Henri Hirvelä

Erikoistehosteiden toteutus mobiilipeleissä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

20.9.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Henri Hirvelä Erikoistehosteiden toteutus mobiilipeleissä 39 sivua + 1 liite 20.9.2018
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	tieto- ja viestintätekniikka
Ammatillinen pääaine	pelisovellukset
Ohjaajat	Lehtori Antti Laiho Toimitusjohtaja Roni Jokinen

Insinöörityössä toteutettiin erikoistehosteet asiakkaan valmisteilla olevaan mobiilipeliin. Erikoistehosteet toteutettiin, jotta pelistä saataisiin helppotajuisempi ja näyttävämpi. Erikoistehosteiden toteutusta käsiteltiin työssä hyvin teknisestä näkökulmasta.

Erikoistehoste on yleisesti määritelty videopeleissä siten, että se on jotain, joka liikkuu, mutta ei ole hahmo. Erikoistehosteen pääasiallinen tarkoitus videopeleissä on viestiä pelaajalle pelin tapahtumia. Informaatio välitetään pelaajalle värein, liikkein ja muodoin.

Erikoistehosteen tuotantoprosessi alkaa yleensä referenssien kartoituksella ja tutkimisella. Referenssejä kerätään muun muassa luonnosta ja muista peleistä. Tämän jälkeen siirrytään luonnosteluun, jossa piirretään luonnos tehosteesta. Luonnoksen avulla saadaan kuva tehosteen pääelementeistä sekä alustava lista tehosteen luomiseen tarvittavista resursseista. Lopuksi tehdään kehitystyö, jossa kannattaa keskittyä erityisesti tekniseen toteutukseen näyttävyyden sijaan, sillä toivottu ulkoasu voi muuttua kehitystyön aikana.

Insinöörityön tuloksena valmistuivat pelin erikoistehosteiden testiversiot. Erikoistehosteiden toteuttamiseen käytetyt tekniikat erosivat toisistaan paljon. Toteutuksissa käytettiin eri tekniikoita, kuten varjostimia, hiukkastehosteita, animaatioita ja reaaliaikaista polygonimallin generointia. Erikoistehosteiden lisäksi insinöörityössä valmistui kattava Bézier-käyrätoteutus, joka on uudelleenkäytettävissä asiakkaan tulevissa projekteissa. Valmiita erikoistehosteita käytetään pelin julkaistavassa versiossa. Tehosteita saatetaan jatkokehittää ja toteuttaa lisää, jos peli saa positiivisen vastaanoton julkaisun jälkeen.

Projektin kokemusten perusteella erikoistehosteiden toteuttaminen vaatii usein sekä teknistä osaamista että taiteellista silmää.

Avainsanat	erikoistehoste, VFX, mobiilipeli, Bézier, varjostin, hiukkastehoste

Author Title Number of Pages Date	Henri Hirvelä Visual effects development in mobile games 39 pages + 1 appendice 20 September 2018
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Game Applications
Instructors	Antti Laiho, Senior Lecturer Roni Jokinen, Chief Executive Officer

Visual effects were developed for the customers game in development. The visual effects were developed to make the game easier to understand and to improve its visuals. The development of these effects is covered from a very technical point of view.

Visual effects in video games are usually defined as something that moves, but something that's not a character. The primary objective of visual effects in games is to transmit information to the player by using colors, shapes and motion.

The production pipeline of visual effects usually starts with collecting and studying references. Best references are collected from nature and other games. After studying references comes the concepting phase, where one draws concept art for the effect. From the concept it's easy to outline main motions of the effect and make an initial list of needed resources. After all this, it's time to start developing the effect. One should always focus on technical implementation before artistic adjustments, because the artistic look of the effect may change during the development of the game.

As a product of the thesis, the demo versions of the visual effects for the game were produced. The techniques used to make the effects varied. For example, shaders, particle effects, animations and real time construction of meshes were used to produce the effects. As a byproduct of the project, a comprehensive Bézier spline implementation was made. The implementation can be used by the customer in their upcoming projects. The made visual effects will be used in the release version of the game. More effects may be developed if the game gets positive feedback after the release.

Based on experiences from the thesis it can be stated, that developing visual effects requires technical knowledge and artistic vision.

Keywords	visual effects, VFX, mobile game, Bézier, shader, particle effects

Sisällys

Lyhenteet

VFX	Visual Effects. Visuaalinen (optinen) erikoistehoste tarkoittaa tehostetta, joka on luotu tietokoneella.
UV	U ja V ovat standardikirjaimet pisteen koordinaateille tekstuuriavaruudessa.
2.5D	D-kirjain tulee englannin kielen sanasta dimension eli ulottuvuus. Kolmiulotteiselta vaikuttava, mutta oikeasti kaksiulotteinen. Tai kolmiulotteisuus, jossa esimerkiksi liike on rajoitettu kahteen ulottuvuuteen.
2D	Kaksiulotteisuus
3D	Kolmiulotteisuus
RGB	Kirjaimet tulevat englannin kielen sanoista red, green ja blue. Värimalli, jossa eri värejä muodostetaan sekoittamalla punaista vihreää ja sinistä valoa.

1 Johdanto

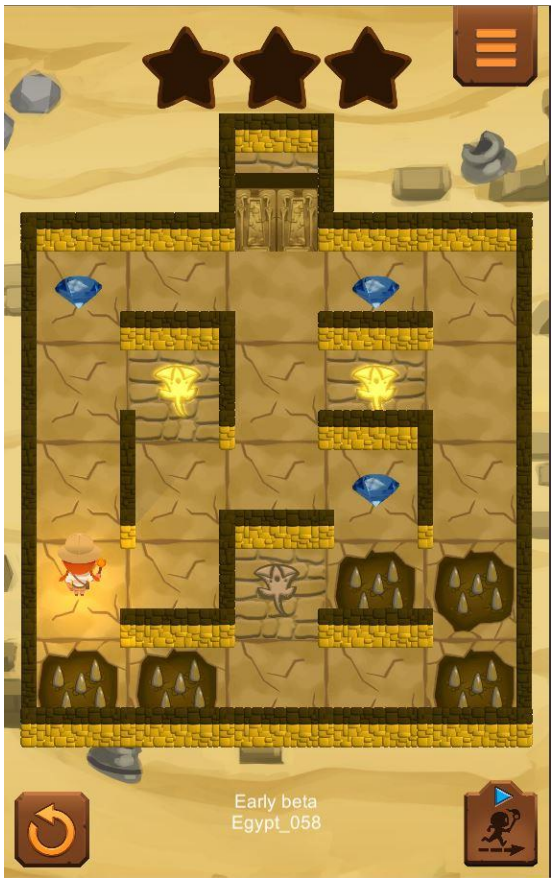
Erikoistehosteet, kuten esimerkiksi räjähdykset, savu, valot ja vesi, ovat peleissä mukana luomassa tunnelmaa ja välittämässä sanatonta informaatiota pelaajalle. Ilman tehosteita suuri osa peleistä vaikuttaisi hyvin elottomilta ja pelaajan olisi vaikeaa ymmärtää pelin interaktioita, kuten vaikka ammutun raketin osumaa ilman räjähdystä. (Romanowska 2017.)

Insinöörityön tavoitteena oli toteuttaa visuaalisia erikoistehosteita Trail of Relics -peliin ja samalla perehtyä tavanomaisimpiin tekniikoihin ja tuotantoprosesseihin, joita videopelien erikoistehostetuotannossa käytetään. Erikoistehosteiden avulla pelistä haluttiin tehdä näyttävämpi ja helppotajuisempi. Insinöörityö tehtiin osana LunarByten Trail of Relics -pelin kehitysprosessia. Pelin päävalikko on nähtävissä kuvassa 1.

Kuva 1. Trail of Relics -pelin päävalikko (2018).

LunarByte on vuonna 2017 perustettu pieni startup-yritys, joka keskittyy mobiilipelituotantoon. Yrityksen ensimmäinen peli, Trail of Relics, on julkaistu rajoitetulle yleisölle Androidilla, sillä se on vielä kehitysvaiheessa.

Trail of Relics on mobiilialustoille Unity-pelimoottorilla kehitetty pulmapeli, jossa pelaaja seikkailee muinaisissa kohteissa etsien aarteita. Pelissä pelaajan tehtävänä on läpäistä sokkelomaisia tasoja ja samalla kerätä mahdollisimman monta timanttia. Tasot koostuvat laatoista, joista suurin osa tekee jotain pelihahmon kävellessä niiden ylitse. Suurin osa laatoista rikkoutuu siten, ettei pelihahmo voi enää kävellä niiden ylitse toistamiseen. Kuviolliset laatat avaavat tason oven, jos niiden kaikkien yli on kävelty, jotta tason voi läpäistä. Kuvassa 2 on pelin kenttä, jossa pelaaja on jo edennyt rikkoen viisi laattaa ja aktivoiden yhden kuviollisen.



Kuva 2. Esimerkkikuva Trail of Relics -pelin pelitilanteesta.

2 Erikoistehosteet videopeleissä

2.1 Erikoistehosteen määritelmä

Elokuvissa erikoistehosteilla tarkoitetaan asioita, joiden oikea toteutus olisi mahdotonta tai liian kallista, kuten esimerkiksi tulivuoren purkaus tai kova väkivalta (Okun & Zwerman 2010: 2). Elokuvateollisuus jakaa erikoistehosteet karkeasti kahteen eri ryhmään: fyysisiin ja visuaalisiin (optisiin) tehosteisiin. Fyysinen erikoistehoste on tehoste, joka voidaan luoda fyysisesti, sijoittaa ympäristöön ja siten kuvata kameralla. Esimerkkejä fyysisistä tehosteista ovat muun muassa pyrotekniikka- (esimerkiksi tulipalo) ja säätehosteet (esimerkiksi sade tai tuuli). (Okun & Zwerman 2010: 2.)

Visuaalisella tehosteella tarkoitetaan erikoistehostetta, joka on luotu digitaalisesti. Yleensä visuaalisista tehosteista käytetään lyhennettä VFX, joka tulee englannin kielen sanoista visual effects. Visuaaliset tehosteet luodaan tietokoneella elokuvan jälkituotantovaiheessa kameralla kuvatun kuvan päälle. Visuaalisilla tehosteilla voidaan saada aikaan lähes mitä vain savusta avaruustaisteluihin. Esimerkiksi kuvassa 3 on käynnissä säätiedotus, jossa visuaalisia tehosteita käytetään reaaliajassa. Juontajan takana oleva vihreä kangas korvataan halutulla kuvalla, joka on tässä tapauksessa sääkartta. (Okun & Zwerman 2010: 2.)

Kuva 3. Esimerkki visuaalisten tehosteiden käytöstä reaaliajassa säätiedotuksessa (Gallahan 2006).

Videopeleissä erikoistehosteelle ei ole yhtä oikeaa määritelmää. Elokuva-alan standardeilla koko videopeli olisi yhtä isoa tehostetta. Tästä syystä pelistudiot määrittelevätkin usein itse, mikä on niiden peleissä teknisesti erikoistehoste ja mikä ei. Kuitenkin yleistettäessä erikoistehosteeksi videopeleissä voidaan mieltää asia, joka liikkuu mutta ei ole hahmo (Hubbell 2014). Hahmoksi mielletään yleensä objekti, jonka mallilla on vähintään yksi luu. Esimerkiksi vilkkuvat valot, kipinät, usva ja räjähdykset luokitellaan yleisesti erikoistehosteiksi videopeleissä.

Usein peleissä, niin kuin elokuvissakin, erikoistehosteet jaetaan vielä karkeasti kahteen eri luokkaan: ympäristö- ja pelaustehosteisiin. Ympäristötehosteet ovat erikoistehosteita, joita on pelaajan ympäristössä pelissä ja joihin pelaaja voi harvoin tekemisillään vaikuttaa. Esimerkiksi sade, joen virtaus, nuotio, pöllyävä hiekka ja heijastukset ovat ympäristötehosteita. Pelaustehosteet ovat erikoistehosteita, jotka luodaan pelaajan tekemisen perusteella, ja ne voivat vaikuttaa itse pelaamiseen. Esimerkiksi erilaiset loitsut ja pelaajan ammusten räjähdykset ovat pelaustehosteita. (Ordoñez 2017.)

2.2 Erikoistehosteiden tarkoitus videopeleissä

Erikoistehosteiden tarkoitus on välittää informaatiota pelaajalle ja näyttää samalla teemaan ja välitettävään informaatioon sopivalta. Tehosteen informaatio koostuu lähes aina kolmesta tekijästä: muodoista, väreistä ja liikkeistä. Esimerkiksi parannustehosteen värejä ovat usein vihreän eri sävyt, sillä pelaajat ovat jo vuosien ajan tottuneet yhdistämään vihreän värin parantamiseen. Kuvassa 4 on Fortnite-pelin parannustehoste käytettäessä lääkintäpakkausta. Muoto kertoo pelaajalle, millä alueella tehosteen vaikutus on. Parannustehoste voisi olla esimerkiksi ympyrän muotoinen, jolloin pelaaja olettaa, että ympyrässä ollessaan hän parantuu. Tehosteen liike kertoo, missä vaiheessa elinkaartaan tehoste on. Esimerkiksi tehosteen värien hiipuessa se on todennäköisesti loppumaisillaan. (Jevremovic 2018; Romanowska 2017.)

Kuva 4. Vihreä parannustehoste pelaajan käyttäessä lääkintäpakkausta Fortnite-pelissä (Fortnite 2017).

Lähes kaikilla tehosteilla on kolme vaihetta elinkaaressaan: enteily, kliimaksi ja häivytys. Enteily on vaihe, jonka on tarkoitus kertoa pelaajalle, minne, milloin ja mitä on tulossa. Kliimaksin on tarkoitus näyttää, että nyt tehoste on käynnissä ja sen efektit tapahtuvat. Häivytys puolestaan on olemassa siksi, että se kertoo efektin loppuvan. Kuvassa 5 näkyvät meteoritehosteen kaikki kolme vaihetta vasemmalta oikealle: enteily, kliimaksi ja häivytys. (Jevremovic 2018.)

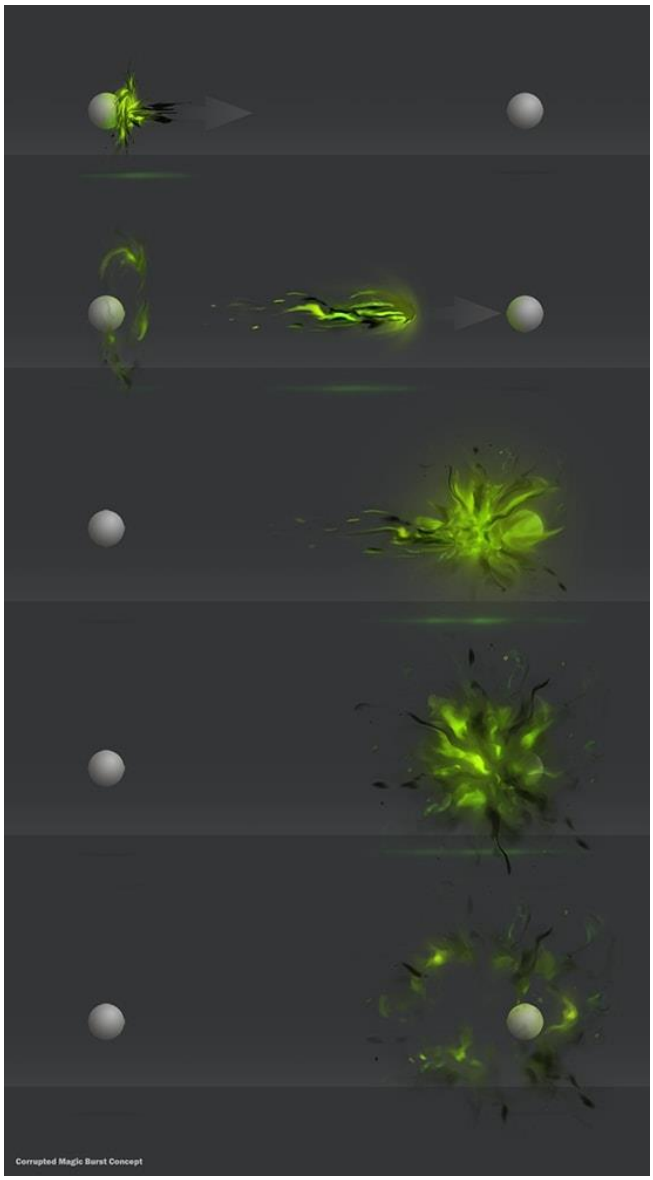
Kuva 5. Meteoritehosteen elinkaaren vaiheet (Jevremovic 2018).

Tehosteen näyttävyys on peleissä aina toissijaista informaation välittämisen rinnalla. Jos informaatio ei välity kunnolla, siitä saattaa seurata väärinkäsitys, joka voi johtaa pelaajan turhautumiseen ja pahimmassa tapauksessa pelaamisen lopettamiseen. Hyviä tapoja viestiä pelaajalle on näyttää jotain, missä on jo jotain tuttua, tai suunnitella tehosteen liike siten, että interaktiot ovat selkeitä. Esimerkiksi valkoinen on yleensä hyvä ja musta paha. (Guerrette 2016.)

2.3 Tuotantoprosessi

Erikoistehosteen tuotantoprosessi alkaa aina etsimällä vastauksia sellaisiin kysymyksiin kuin: Kuinka kompleksi tehoste on kyseessä? Mikä tehosteen merkitys on pelissä? Minkälainen tyyli tehosteella on? Kun kysymyksiin on saatu alustavat vastaukset, aletaan kerätä referenssejä. Referenssejä kerätään, jotta niistä voidaan saada inspiraatiota tehosteen ulkoasulle ja tekniselle toteutukselle. Referenssejä kannattaa etsiä peleistä ja erityisesti luonnosta. Oikeaa elämää ja luontoa tutkimalla saa usein parhaat referenssit realistisille tehosteille, kuten nuotiolle tai nesteiden liikkeille. (de Laat 2018; Romanowska 2017.)

Kun referensseihin ollaan tyytyväisiä ja tehosteen perusrakenne on selvillä, siirrytään luonnosteluun. Luonnostelussa keskitytään tehosteen elinkaaren pääpiirteisiin ja tapahtumiin, jotka ovat olennaisia tehosteelle. Huolellisen luonnostelun avulla saadaan käsitys tehosteen toteutuksen laajuudesta sekä alustava lista tarvittavista resursseista. Kuva 6 on korruptoituneen loitsun luonnos, josta nähdään tehosteen päävaiheet. Kuvasta voidaan myös päätellä, että tarvittavia resursseja ovat ainakin muutama erilainen tekstuuri, pari erimuotoista polygonimallia, jonkinlainen valo ja pari animaatiota. (de Laat 2018.)



Kuva 6. Erikoistehosteen luonnostaidetta (de Laat 2018).

Kun tehosteen luonnokseen ollaan tyytyväisiä, voidaan aloittaa kehitystyö. Alkuvaiheessa tehosteen näyttävyyteen ei ole syytä panostaa paljon, koska pelin visuaalinen tyyli muuttuu usein pelin kehitystyön aikana. Alkuvaiheessa on syytä panostaa tekniseen toimivuuteen ja selkeyteen, jotta tehostetta ei tarvitsisi tehdä kokonaan uudestaan visuaalisen tyylin muuttuessa. Kehitystyön aikana tehosteiden optimointi tulee pitää mielessä koko ajan, sillä käytettävää tekniikkaakin saattaa joutua vaihtamaan, jos jokin tehoste vaatii esimerkiksi liikaa laskentatehoa. Erikoistehosteiden pitää näyttää hyvältä ja toimia hyvin eri suunnista katsottuna, erilaisissa valaistuksissa ja eritehoisilla laitteilla. (Romanowska 2017.)

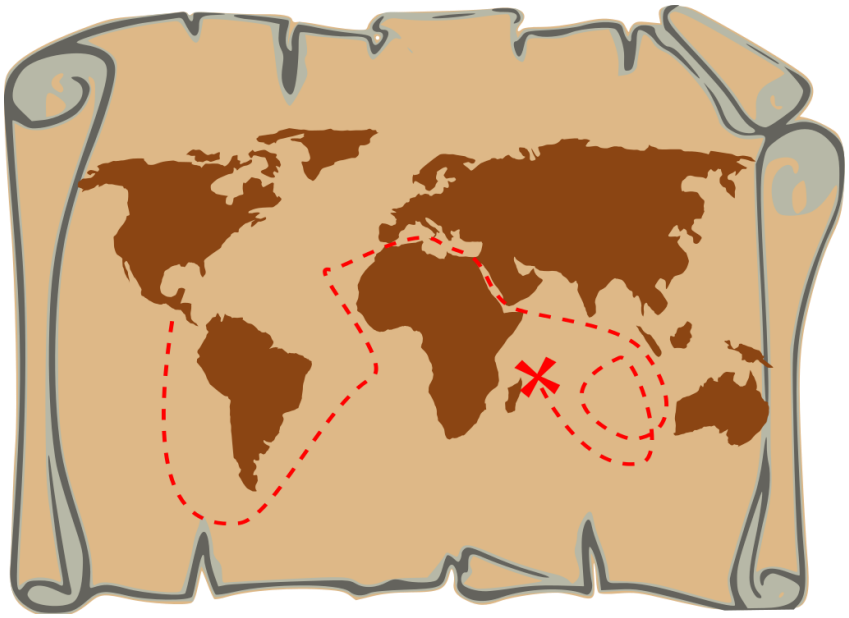
3 Referenssitehosteet

3.1 Reittitehoste

Etenkin strategiapeleissä reittitehosteet ovat yleisiä. Esimerkiksi kuvassa 7 on peli Europa Universalis IV pelitilanne, jossa punainen nuoli näyttää pelaajalle hänen sotajoukkojensa reitin yksityiskohtaisesti. Reitin näyttäminen on tärkeää, jotta pelaajan ei tarvitse arvailla tekoälyn kulkemaa reittiä tai muistaa piirtämäänsä reittiä ulkoa.

Kuva 7. Punainen reittitehoste pelissä Europa Universalis IV. (Europa Universalis IV 2013)

Taiteelliselta ilmeeltään insinöörityössä työstetyn Trail of Relics -pelin reittitehosteen haluttiin muistuttavan tavanomaisen aarrekartan katkoviivaa. Kuvassa 8 on kuva perinteikkäästä aarrekartasta, joka on monille tuttu populaarikulttuurista. Tällainen katkoviiva on monille tuttu jo ennestään, ja se on helppo erottaa pelin muusta grafiikasta.



Kuva 8. Tavanomainen aarrekartta punaisella katkoviivalla ja rastilla (Treasure map 2010).

Trail of Relics -pelissä pelaaja piirtää hahmon reitin sokkelon läpi, ja jottei pelaajan tarvitsisi muistaa ulkoa piirtämäänsä reittiä, peliin päätettiin toteuttaa tehoste, joka havainnollistaa pelaajalle siihen asti piirretyn reitin. Reittitehoste suunniteltiin käytettäväksi myös pelin päävalikossa, jossa se havainnollistaa pelihahmon matkaa pelin kohteiden välillä.

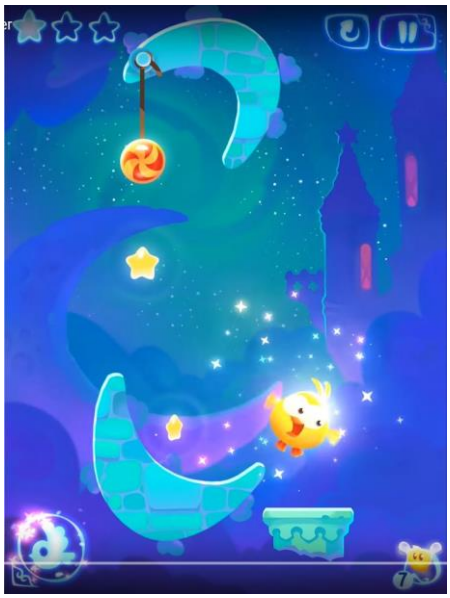
3.2 Keräilytehoste

Trail of Relics -pelin pelitesteissä oli saatu selville, etteivät pelaajat ymmärtäneet kerättävien timanttien vaikutusta pelissä. Pelin jokaisessa tasossa on kolme timanttia, joita keräämällä pelaaja ansaitsee valuuttaa ja saa paremman arvosanan tasosta. Arvosana viestitään pelaajalle tähtinä yhdestä kolmeen. Timantin keräystehosteen tulisi erityisesti korostaa timanttien ja tähtien yhteyttä pelaajalle.

Tehosteita kerättäville esineille on melkeinpä jokaisessa pelissä, jossa kerättäviä esineitä on. Esimerkiksi peleissä Dota 2 ja Cut the Rope: Magic on keräilytehosteet. Dota 2 -pelin tasossa on kerättäviä riimuja, joista palkkioriimu antaa kerääjälle kultaa. Kuvasta 9 nähdään, kuinka saadun kullan määrää havainnollistetaan tehosteella. Tehosteessa hahmon kohdalla lentelee kultakolikoita ja saatu rahasumma näkyy pelihahmon yläpuolella. Cut the Rope: Magic -pelin tasoissa on kerättäviä tähtiä. Kuvassa 10 näkyy tähden

keräämistä seuraava ilotulitusmainen tehoste ja tähtikuvion täyttyminen kuvaruudun vasemmassa ylänurkassa. Näiden keräilytehosteiden kohdalla voidaan olla miltei varmoja, että ne on toteutettu hiukkastehosteiden avulla. Kuvan 9 keltainen teksti ”+100” ei tosin varmasti ole osa hiukkastehostetta, vaan se on todennäköisesti jonkinlainen animaatio.

Kuva 9. Kerätyn palkkioriimun tehoste pelissä Dota 2 (Dota 2 2013).

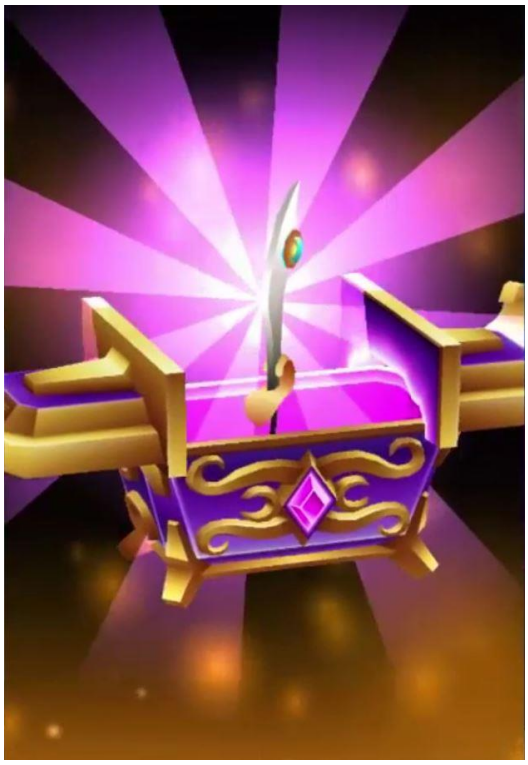


Kuva 10. Kerätyn tähden tehoste pelissä Cut the Rope: Magic (Cut the Rope: Magic Release Trailer 2015).

3.3 Palkintoruudun tehoste

Mobiilipelit ovat usein ilmaisia, joten niiden rahoitus on saatava muualta kuin pelin ostotapahtumasta. Yleisimpiä keinoja ovat pelin sisäiset ostokset ja mainokset. On siis erityisen tärkeää tehdä ostoksien tekemisestä ja mainosten katselusta houkuttelevia.

Trail of Relics -pelissä tulee silloin tällöin tason läpäisyn jälkeen palkintoruutu. Palkintoruudussa pelaajan on mahdollista katsoa noin 30 sekunnin mittainen mainos palkintoa vastaan. Palkinnon hienoutta haluttiin korostaa erikoistehosteella, jotta useammat katsoisivat mainoksen. Kuvassa 11 on pelin Blades of Brim palkintoruutu, jossa pelaaja on avannut palkintona saamansa arkun. Kuva ei kerro kaikkea, mitä tehosteessa tapahtuu. Tehosteen eri vaiheet on todennäköisesti luotu käyttämällä erimuotoisia polygonimalleja, ja, manipuloimalla niitä ja niiden tekstuureja.



Kuva 11. Palkintoarkun avaaminen pelissä Blades of Brim (Blades of Brim 2016).

3.4 Hiekkamyrsky

Trail of Relics -pelin alussa pelaajan täytyy joka kerta valita pelin kartalta kohde, jota hän haluaa tutkia. Pelin alussa suurin osa kohteista on kuitenkin suljettu pelaajalta. Kohteiden suljettua tilaa haluttiin viestiä pelaajalle erikoistehosteen avulla.

Tehosteen taiteelliseksi ilmeeksi päätettiin hiekkamyrsky. Pelissä pelaaja seikkailee muinaisen Egyptin kohteita tutkien, joten hiekkamyrsky tuntui luontevalta teemalta tehosteelle. Jotta tehoste tuntuisi hiekkamyrskyltä, sen täytyisi peittää näkymää hieman, kuitenkaan häiritsemättä pelaamista, ja liikkua koko ajan. Kuvassa 12 on oikea hiekkamyrsky Gizan pyramideilla. Kuvassa 13 on hiekkamyrskytehoste pelistä Hearthstone ja se lienee toteutettu liikkuvan tekstuurin avulla.

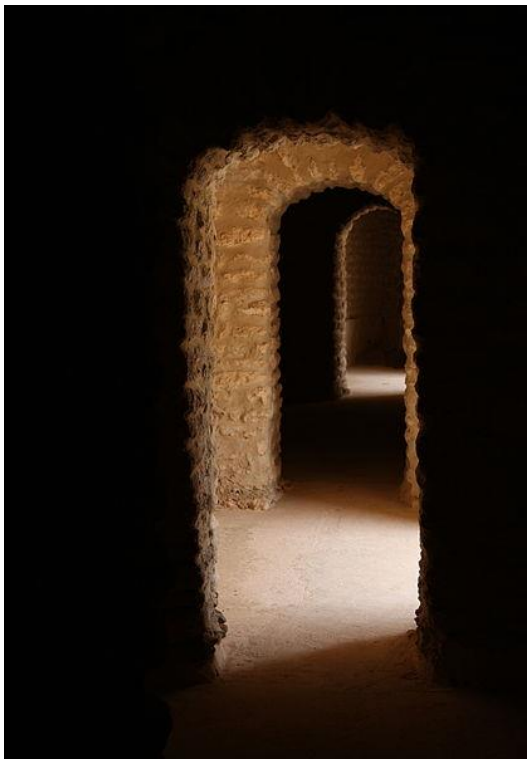
Kuva 12. Oikea hiekkamyrsky Gizan pyramideilla (Hiekkamyrsky).

Kuva 13. Hiekkamyrskytehoste pelistä Hearthstone (Hearthstone 2014).

3.5 Valaistus

Valaistusta ei pelien tuotannossa yleensä lueta erikoistehosteeksi, vaan kokonaan omaksi osa-alueekseen. Peleissä valaistuksella voidaan luoda eri tunnelmia, täydentää erikoistehosteita ja tehostaa kolmiulotteisuuden tunnetta (Abdillah 2016). Mobiilipeleissä harvoin näkee kompleksia valaistusta niiden suppeuden ja mobiililaitteiden heikon tehon vuoksi (Frisvad ym.: 1). Tästä syystä mobiilipelien kehitystiimeissä ei usein ole valaistukselle omistautunutta kehittäjää.

Trail of Relics -pelissä valaistuksella haluttiin luoda tunnelmaa kolmiulotteisuudesta ja hahmon pitelemästä soihdusta maanalaisissa kammioissa. Kuvassa 14 on pelin valaistuksella tavoiteltua tunnelmaa. Valaistus ei välitä Trail of Relics -pelissä informaatiota, vaan on puhtaasti taiteellinen tehoste.



Kuva 14. Esimerkki tavoitellusta valaistuksesta peliin (Passage souterrain à Oudna 2014).

4 Tehosteiden toteutus

Insinöörityössä kaikkia tehosteita toteutettaessa täytyi pitää mielessä, että peliä kehitettiin mobiilialustoille, joilla on kosketusnäyttö. Mobiililaitteita on hyvin eri tehoisia ja kokoisia. Erityisesti tehosteiden laskentatehovaatimukset täytyi pyrkiä pitämään alhaisina, jotta peli toimisi mahdollisimman monella eri laitteella. Mobiililaitteiden näytöt ovat usein pieniä, minkä takia tehosteista täytyi tehdä riittävän isoja. Liian pienet tehosteet voisivat jäädä pelaajalta huomaamatta, tai ne voisivat jäädä piilon pelaajan sormien alle.

Huomioitavaa on, että työssä käsitellään yksityiskohtaisesti vain työn erikoistehosteiden tekemiseen käytettyjä ja yleisimpiä erikoistehosteiden toteutusmenetelmiä. Pelien erikoistehosteiden tekemiseen on olemassa monia muitakin tapoja ja ohjelmistoja kuin työssä käsitellyt. Tavallisimpia toteutusmenetelmiä, joita työssä ei päästy käyttämään, ovat muun muassa tekstuurisarja-animaatiot ja virtaustekniikat.

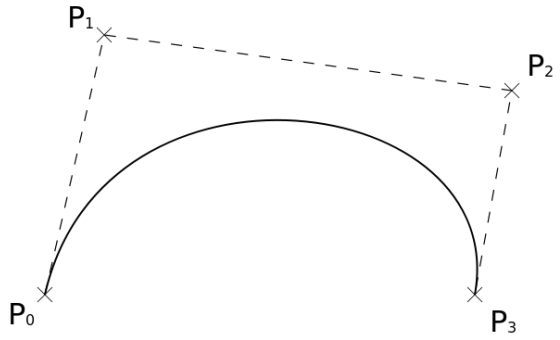
4.1 Reittitehoste

Reittitehosteen vaatimukset olivat selkeitä. Tehosteen tulisi piirtyä pelaajan piirtämän reitin mukaan, ja sen pitäisi poistua pelihahmon kävellessä sitä pitkin. Reitin käännekohdissa ei myöskään saisi olla teräviä kulmia, ja tehosteen grafiikan pitäisi näyttää yhtenäiseltä.

Aluksi tehostetta koetettiin toteuttaa Unity-pelimoottorin viivarenderoija-komponentilla. Viivarenderoija ottaa parametrina taulukon kolmiulotteisia pisteitä ja piirtää suoran viivan, joka kulkee jokaisen pisteen kautta (Unity Manual – Line Renderer 2017). Tästä toteutustavasta kuitenkin luovuttiin pian, kun huomattiin, että viivarenderoijalla mutkien piirrosta olisi tullut hyvin työlästä. Haluttaessa piirtää loiva kaari olisi jouduttu laskemaan kaarelta niin monta pistettä, että viivan kulkiessa niiden kautta se olisi näyttänyt kaarelta kauempaa katsottaessa. Olisi helpompaa ja nopeampaa toteuttaa tehoste muulla tavoin.

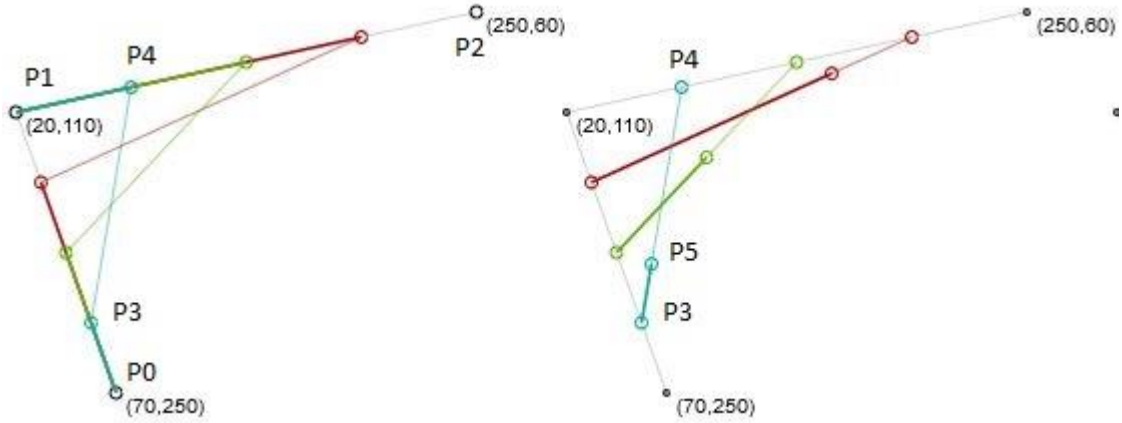
Reittitehoste päätettiin lopuksi toteuttaa Bézier-käyrän avulla. Bézier-käyrä on piirtotekniikka, jossa vähintään kahden pisteen avulla piirretään pisteiden välille käyrä. Keskeistä tekniikassa ovat pisteiden kahvat, joita siirtämällä käyrää saadaan muokattua. Kuvassa 15 on esimerkki kuutiollisesta Bézier-käyrästä, joka on piirretty pisteiden P0 ja P3 välille kahvapisteiden P1 ja P2 avulla. Yleensä tietokoneohjelmissa käytetään kvadraattisia tai

kuutiollisia Bézier-käyriä. Kvadraatti ja kuutio viittaavat siihen, kuinka montaa pistettä käyrän piirtämisessä käytetään hyväksi. Useampia pisteitä ei tietokoneohjelmissa usein käytetä kompleksisten käyrien piirtoon, sillä ne lisäävät laskentatehon tarvetta huomattavasti. Kompleksisempia käyriä voidaan saada aikaan ilman ylimääräisiä pisteitä yhdistämällä yksinkertaisempia käyriä toisiinsa. Unity-pelimoottorissa ei ole valmista Bézier-käyrätoteutusta, joten toteutus täytyi tehdä itse. (Xuexiang & Junxiao 2014.)



Kuva 15. Kuutiollinen Bézier-käyrä (Bézier-käyrä 2006).

Bézier-käyrä on lineaarisen interpoloinnin tulos. Kuva 16 havainnollistaa kvadraattisen Bézier-käyrän piirtoa vaiheittain 25 prosenttiyksikön tarkkuudella. Kuvan 16 käyrän alkupisteiden (P0, P1 ja P2) etäisyydet toisistaan tiedetään, joten niiden väleille voidaan laskea uudet pisteet halutuin välein syötteen avulla, eli interpoloida. Tästä saadaan pisteet P3 ja P4, joiden välille voidaan laskea piste P5 samalla tavalla. Kun kaikki mahdolliset interpoloinnit on tehty syötteellä ja yksi piste on enää jäljellä (P5), tiedetään pisteen olevan käyrällä. Näitä käyrällä sijaitsevia pisteitä interpoloidaan niin tiheästi, että ollaan tyytyväisiä käyrän muotoon. Lopuksi käyrä piirretään siten, että se kulkee pisteestä P0 kaikkien käyrälle interpoloitujen pisteiden kautta pisteeseen P2. Esimerkiksi kuvan 16 valmiista käyrästä tulee melko kulmikas siksi, että se laskettiin pienellä, 25 prosenttiyksikön tarkkuudella. (Kamermans 2011.)



Kuva 16. Kvadraattisen Bézier-käyrän piirto vaiheittain pisteiden P0 ja P2 välille (Kamermans 2011).

Koodissa ei tarvitse laskea jokaista välipistettä, kuten äskeisessä esimerkissä. Reittitehostetta varten toteutettiin kuutiollisen Bézier-käyrän implementaatio, joka käyttää hyväkseen kaavaa 1. Kaavaan sijoitetaan alkupisteet ja syöte, josta saadaan piste käyrällä. Kuutiollisella Bézier-käyrällä saatiin piirrettyä monipuolisempia käyriä kuin kvadraattisella käyrällä, ilman suurta eroa laskenta-ajassa. (Flick 2014.)

$$B(t) = (1-t)^3 P0 + 3(1-t)^2 t P1 + 3(1-t)t^2 P2 + t^3 P3 \quad (1)$$

t on syöte $(0 \leq t \leq 1)$
$P0$ on käyrän alkupiste
$P1$ on käyrän ensimmäinen kahvapiste
$P2$ on käyrän toinen kahvapiste
$P3$ on käyrän loppupiste

Implementaatiossa pystytään liittämään kuutiollisia Bézier-käyriä toisiinsa kompleksisemmiksi yhdistelmäkäyriksi. Yhdistelmäkäyrä koostuu kahdesta tai useammasta käyrästä, joita käytetään siten, että edellisen käyrän loppupiste on seuraavan alkupiste. Reittitehoste on toteutettu yhdistelmäkäyränä. (Flick 2014.)

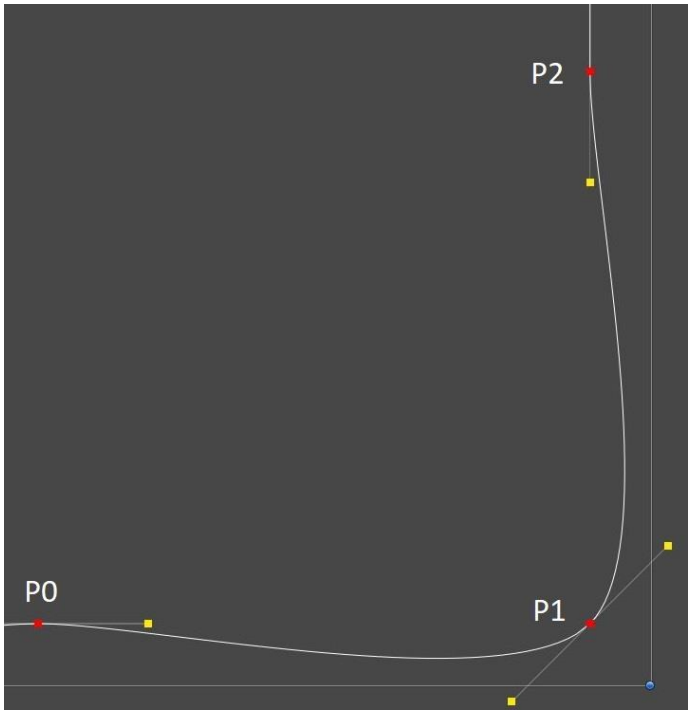
Reittitehosteen käyrien alku- ja loppupisteet saadaan pelaajan syötteestä tämän piirtäessä reittiä tason läpi. Pelaajan antaessa sallitun syötteen jollekin tason laatalle luodaan uusi käyrä. Uuden käyrän alkupiste sijoitetaan pelihahmon kohdalle, jos käyrä on ensimmäinen piirrettävä, muutoin se sijoitetaan edellisen käyrän loppupisteen kohdalle. Uuden käyrän loppupiste sijoitetaan pelaajan syötteen osoittaman laatan keskelle.

Kahvapisteet sijoitetaan poikkeavasti. Kun käyrä lisätään yhdistelmäkäyrään, uuden käyrän ensimmäinen kahvapiste sijoitetaan kaavan 2 avulla, ja edellisen käyrän jälkimmäinen kahvapiste kaavan 3 avulla. Edellisen käyrän jälkimmäisen kahvapisteen sijainti on olennaista laskea uudelleen, jotta saadaan aikaan miellyttävän näköisiä mutkia ilman teräviä kulmia. Uuden käyrän toinen kahvapiste sijoitetaan samaan linjaan tämän alku- ja loppupisteen kanssa, jotta se on suorassa. Kuvassa 17 on Unity-pelimoottorissa piirretty kolmen kuutiollisen Bézier-käyrän yhdistelmäkäyrä.

$$B(s) = P1 + \left(\widehat{P2} - \widehat{P0}\right) \cdot \left\|\overrightarrow{P1P2}\right\| \cdot s \quad (2)$$

$$B(s) = P1 + \left(\widehat{P0} - \widehat{P2}\right) \cdot \left\|\overrightarrow{P1P0}\right\| \cdot s \quad (3)$$

$P0$ on edellisen käyrän alkupiste
$P1$ on uuden käyrän alkupiste
$P2$ on uuden käyrän loppupiste
s on automaattisen asetuksen haluttu vahvuus



Kuva 17. Unity-pelimoottorilla piirretty kuutiollisista Bézier-käyristä koostuva yhdistelmäkäyrä. Keltaiset pisteet ovat kahvapisteitä ja punaiset ovat alku- ja/tai loppupisteitä.

Reittitehostetta varten täytyi vielä luoda polygonimalli yhdistelmäkäyrän avulla, jotta tehosteen varsinainen katkoviivagrafiikka saadaan piirrettyä. Ensimmäinen askel polygonimallin luonnissa on laskea yhdistelmäkäyrälle pisteitä tietyn välimatkan välein. Pisteet

yhdistelmäkäyrälle lasketaan liitteen 1 ohjelmakoodin kuvaamalla tavalla. Välimatkan pituus on valittava tarkkaan, sillä mitä pienempi välimatka on, sitä tarkempi on malli. Tarkempi malli vaatii enemmän välimuistia ja sen luonti enemmän laskentatehoa. On tärkeää löytää tasapaino mallin tarkkuuden ja vaaditun laskentatehon välillä, jotta reittitehoste toimii halutulla tavalla mahdollisimman monella laitteella. (Lague 2018.)

Seuraavaksi lasketaan polygonimallin kärkipisteet tasaisin välimatkoin käyrällä sijaitsevien pisteiden avulla. Jokaisen pisteen kohdalla lasketaan kaksi kärkipistettä. Pisteelle lasketaan ”eteenpäin” osoittava yksikkövektori kaavalla 4, jonka ristitulosta Unity-pelimoottorin z-akselin suuntaisen yksikkövektorin kanssa saadaan ”oikea” kärkipiste (kaava 5). ”Vasen” kärkipiste puolestaan on ”oikean” kärkipisteen vastavektori. Saadut kärkipisteet asetetaan järjestyksessä (vasen, oikea, vasen…) listaan. (Lague 2018.)

$$\hat{f} = (P1 - P0 + P0 - P2) \div |(P1 - P0 + P0 - P2)| \quad (4)$$

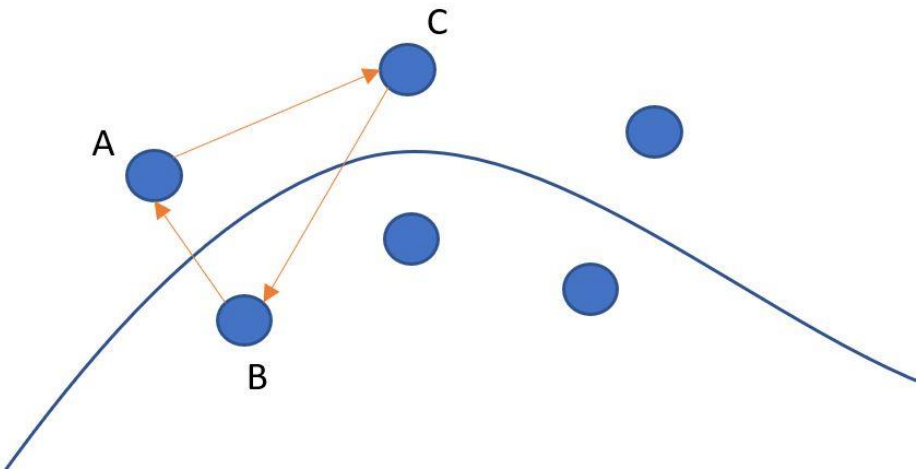
$P0$ on piste, jolle eteenpäin vektoria lasketaan
$P1$ on edellinen piste
$P2$ on seuraava piste

$$x = P0 + (\hat{f} \times \hat{k} \cdot W \cdot 0{,}5) \quad (5)$$

$P0$ on kaavan 4 $P0$
$\hat{f}$ on kaavan 4 tulos
$\hat{k}$ on Unity-pelimoottorin koordinaatiston z-akselin suuntainen yksikkövektori
W on polygonimallin haluttu leveys

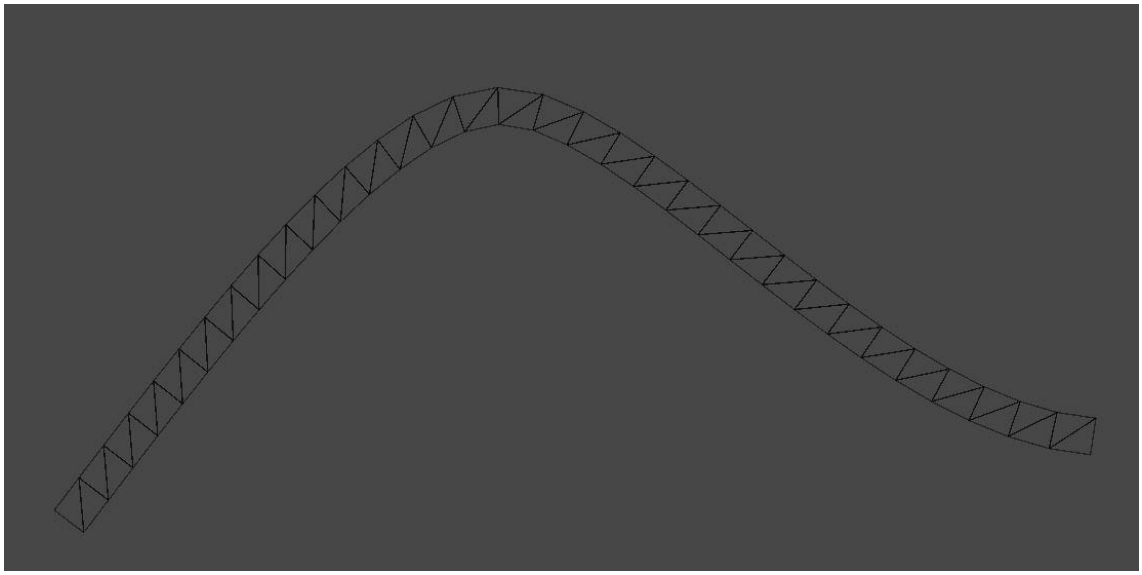
Kaavaan 4 on olemassa kaksi poikkeustapausta: toinen laskettaessa ensimmäiselle pisteelle ja toinen laskettaessa viimeiselle pisteelle, sillä molemmista puuttuu joko seuraava tai edellinen piste. Näissä tapauksissa kaavasta yksinkertaisesti jätetään pois molemmat erotuksista, joissa puuttuvaa pistettä käytettäisiin.

Laskettujen kärkipisteiden järjestys tiedetään, joten niiden väleille voidaan muodostaa kolmioita kuvan 18 osoittamalla tavalla.



Kuva 18. Yksinkertaistettu esimerkki kolmion muodostamisesta kärkipisteiden A, B ja C välille.

Kolmiot muodostetaan aina yhdistämällä pisteitä myötäpäivään. Tällöin kolmion pinnan normaali on oletuksella kameraa kohti, jolloin polygonimallin avulla piirretty kuva näkyy ilman, että mallia tarvitsee kääntää. Kun kaikki pisteet yhdistetään tällä tavoin, saadaan aikaan kolmioverkko, kuten kuvassa 19.



Kuva 19. Yhdistelmä-Bézier-käyrän avulla luotu kolmioverkko.

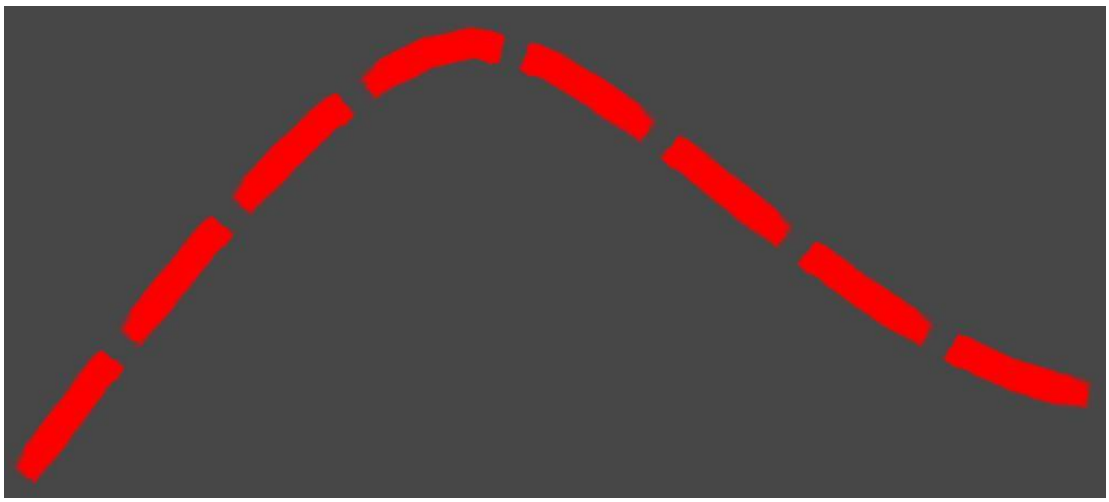
Jokaiselle kärkipisteelle lasketaan UV-koordinaatit. Ne kuvaavat mallia kaksiulotteisesti ja määräävät tekstuurin sijainnin polygonimallilla. U-koordinaatti on x-akselin suuntainen

ja V-koordinaatti vastaavasti y-akselin suuntainen komponentti. Käyrän polygonimalli on yksinkertainen, ja tekstuuri piirretään aina mallin laidasta laitaan, joten U-koordinaatti on aina ”vasemmalla” kärkipisteellä 0 ja ”oikealla” 1. V-koordinaatti kasvaa mallin alusta loppuun nollasta yhteen. Jokaisen kärkipisteparin molemmilla pisteillä on sama V-koordinaatti, joka lasketaan kaavalla 6. (Lague 2018.)

$$V = 1 - |2 \cdot (i : (n-1)) - 1| \quad (6)$$

i on kärkipisteparin järjestysnumero (0…n)
n on kärkipisteparien määrä

Polygonimalli on nyt valmis, ja reittitehoste pystytään piirtämään. Kuvassa 20 on kuvan 19 polygonimallin avulla piirretty reittitehoste. Tehosteen tekstuurista täytyi tehdä saumattomasti toistuva, jotta se ei näyttäisi rikkonaiselta. Tekstuurin toistuvuus täytyi säätää peliin sopivaksi, jotta se näyttäisi miellyttävältä eikä esimerkiksi venytetyltä tai litistetyltä.



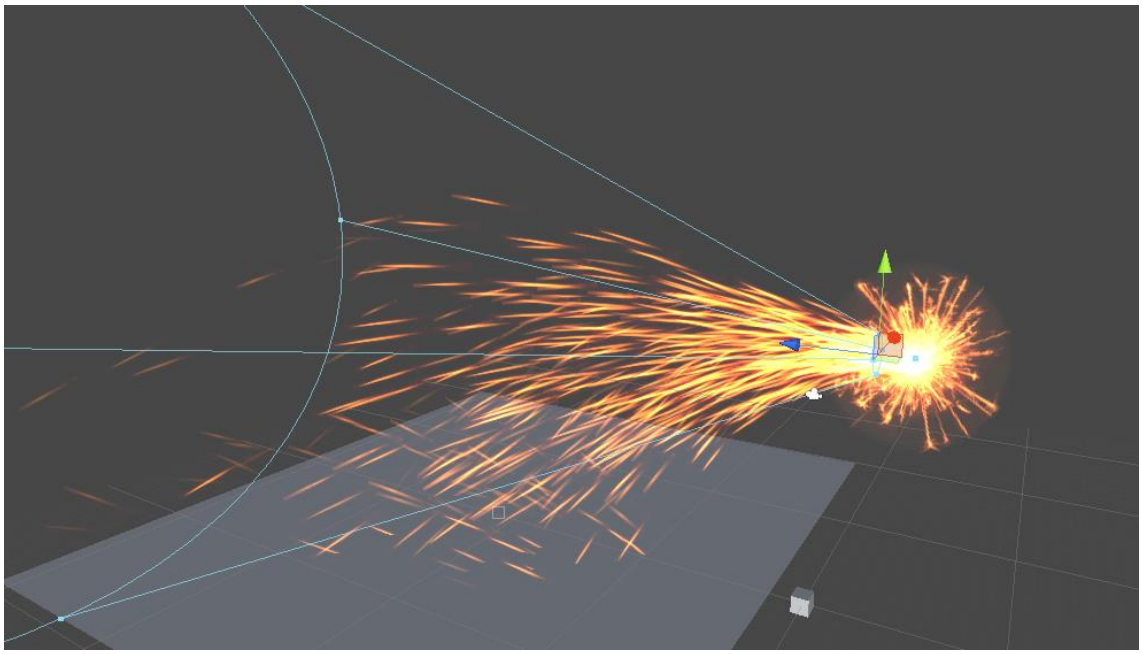
Kuva 20. Valmis reittitehoste.

Yksi reittitehosteen vaatimuksista oli, että grafiikan täytyy poistua pelihahmon kävellessä sen yli. Ensin grafiikan poisto yritettiin toteuttaa siten, että käyrän alkupistettä siirretään pelihahmon mukana sen liikkuessa. Tällä tavalla ei kuitenkaan saatu aikaan halutun mukaista tehostetta. Alkupisteen siirtäminen aiheutti koko käyrän uudelleen laskemisen jokaisella ruudunpäivityksellä, mikä sai grafiikan ”liukumaan”. Grafiikan poisto saatiin lopulta toimimaan muokkaamalla reittitehosteen polygonimallia hahmon liikkuessa. Ensin etsitään hahmoa lähimpänä oleva kärkipistepari polygonimallista. Kärkipisteiden järjestys polygonimallissa tiedetään, ja ensimmäiset kärkipisteparit muodostavat mallin ensimmäiset kolmiot. Esimerkiksi jos pelaajahahmo on lähimpänä toista kärkipisteparia

reittitehosteen polygonimallin kolmiolistasta voidaan poistaa ne kolmiot, joiden muodostamiseen oli käytetty ensimmäistä kärkipisteparia. Seuraavalla ruudunpäivityksellä piirtyy kahta polygonia lyhyempi reittitehoste.

4.2 Hiukkastehosteet

Hiukkastehoste tarkoittaa joukkoa pieniä ja yksinkertaisia kappaleita – hiukkasia, jotka ovat osa hiukkasjärjestelmää. Järjestelmä luo hiukkasia muotonsa sisältä tai sen pinnalta. Hiukkasjärjestelmissä on usein suuri määrä ominaisuuksia, joita muokkaamalla tehosteesta saadaan halutun näköinen. Esimerkiksi Unity-pelimoottorin hiukkasjärjestelmässä muun muassa koko, nopeus, painovoima, tuuli, valot, värit ja törmäykset ovat mahdollisia ominaisuuksia. Yksinkertaisimmillaan hiukkanen voi olla piste, jolla on paikka, rotaatio ja mittakaava ja jonka mukaan esimerkiksi kuvaa tai valoa liikutetaan pelissä. Hiukkasjärjestelmillä toteutetaan peleissä usein tehosteita, joiden vastineet luonnossa ovat hyvin monimutkaisia ja/tai joita olisi erittäin vaikeaa esittää kuvilla tai malleilla. Esimerkiksi nesteet, loitsut, savu ja liekit ovat tällaisia tehosteita. Useissa Trail of Relics -pelin erikoistehosteissa hyödynnettiin Unity-pelimoottorin hiukkasjärjestelmää sen monipuolisuuden ja helppokäyttöisyyden ansiosta. Kuvan 21 kipinätehoste on luotu Unityn hiukkasjärjestelmällä. (Unity Manual – What is a Particle System.)



Kuva 21. Unity-pelimoottorin hiukkasjärjestelmällä luotu kipinätehoste (Heazlewood 2013).

Timantin keräystehoste

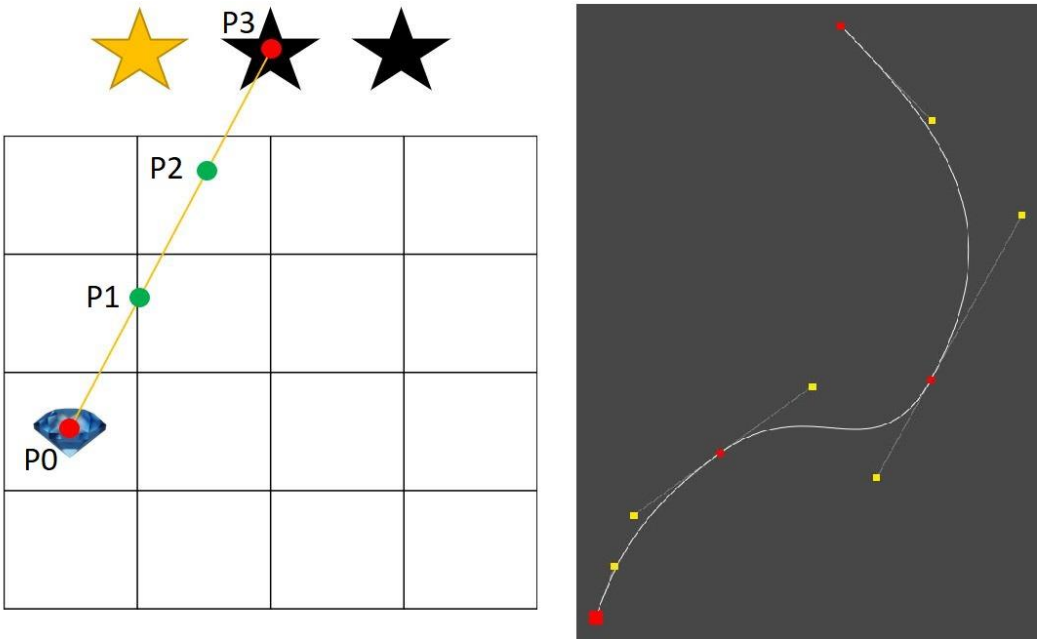
Trail of Relics -pelin pelitesteissä pelaajat eivät yleensä ymmärtäneet timanttien yhteyttä kentän läpäisystä saataviin tähtiin. Kun pelaaja keräsi timantin pelin pelitestiversiossa, kuvan ylälaitaan vain ilmestyi tähti sille tarkoitettuun muottiin. Tähden ilmestyminen jäi monilta huomaamatta, mikä johti ongelmiin pelin pisteytyslogiikan ymmärtämisessä.

Tärkeintä tehosteen toteutuksessa oli viestiä pelaajalle kerätyn timantin yhteys kentän läpäisystä saataviin tähtiin. Tehosteen näyttävyys oli toissijaista mutta silti tärkeää, jotta timantit olisivat houkuttelevampia pelaajille. Parhaaksi tavaksi todettiin viestiä tähtien ja timanttien yhteys tehosteella, joka lentäisi kerätyn timantin kohdalta tähden muottiin ja aktivoisi tähden.

Tehoste päätettiin toteuttaa hiukkasjärjestelmillä, joita liikutettaisiin Bézier-käyrää pitkin. Bézier-käyrätoteutus on kuvattu yksityiskohtaisesti luvussa 4.1. Tällä tavalla tehosteen liikerataan on täysi kontrolli, ja se on helppo määrittää. Näyttävyyttä ja variaatiota tehosteeseen saatiin hiukkasjärjestelmän satunnaismuuttujilla sekä mutkikkaalla Bézier-käyrällä.

Tehoste on tarkoitettu liikkumaan, joten ensiksi oli oleellista toteuttaa tehosteen liikkuminen ja liikerata. Paikoillaan ja liikkuessaan tehoste olisi aivan erinäköinen, eikä sen näyttävyyttä voitaisi arvioida kunnolla ennen liikkeen toimimista.

Pelaajahahmon törmätessä timanttiin muodostetaan yhdistelmä-Bézier-käyrä. Yhdistelmäkäyrä koostuu kolmesta kuutiollisesta Bézier-käyrästä, joiden kahvapisteet sijoitetaan automaattisesti luvussa 4.1 esitetyllä tavalla. Yhdistelmäkäyrän ensimmäinen alkupiste, P0, on aina kerätyn timantin kohdalla ja viimeinen loppupiste, P3, aina seuraavaksi aktivoitavan tähden muotin kohdalla. Kaksi muuta käyrien alku- ja loppupistettä, P1 ja P2, sijoitetaan ensin kaavojen 6 ja 7 avulla pisteiden P0 ja P3 väliin. Lopuksi siirretään pisteitä P1 ja P2 satunnaisesti hieman sivuun, jotta saadaan aikaan mutkia. Kuva 22 havainnollistaa käyrän muodostusprosessia.



Kuva 22. Esimerkki keräystehosteen liikeradan muodostamisesta. Oikealla Unity-pelimoottorissa muodostetussa käyrässä punaiset pisteet ovat alhaalta ylöspäin P0, P1, P2 ja P3. Keltaiset pisteet ovat kahvapisteitä.

$$P1 = \overrightarrow{P0P3} \cdot \frac{1}{3} \quad (6)$$

$$P2 = \overrightarrow{P0P3} \cdot \frac{2}{3} \quad (7)$$

$P0$ on yhdistelmäkäyrän alkupiste
$P1$ on yhdistelmäkäyrän toisen käyrän alkupiste
$P2$ on yhdistelmäkäyrän kolmannen käyrän alkupiste
$P3$ on yhdistelmäkäyrän loppupiste

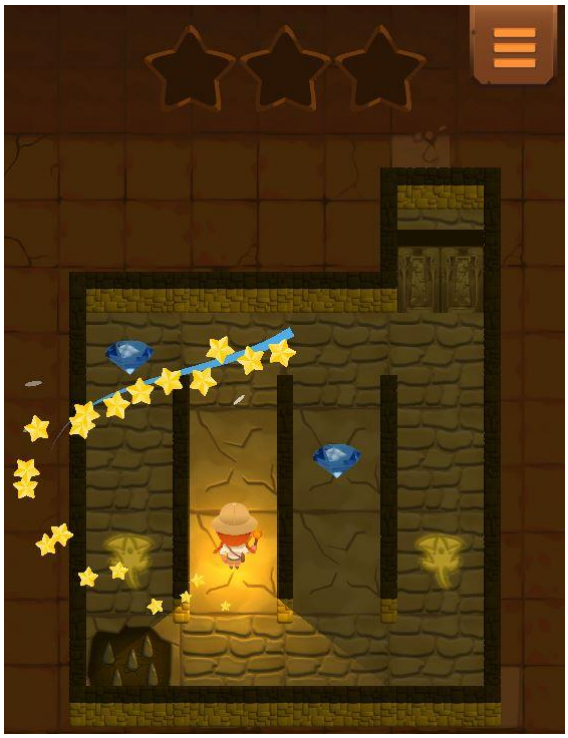
Tehosteen liike käyrää pitkin toteutettiin Bézier-käyrätoteutuksen liikekomponentilla. Liikekomponentin avulla on mahdollista liikuttaa mitä tahansa objektia käyrää pitkin. Liike voi olla yksisuuntaista, toistuvaa tai edestakaista. Komponentti toimii siten, että jokaisella ruudunpäivityksellä lasketaan käyrältä uusi piste, johon liikuteltava objekti siirretään. Keräilytehosteen piste määräytyy kaavan 8 avulla. Liike loppuu, kun kaavan tulos on 1 tai suurempi. Liikuteltava objekti saadaan orientoitua liikesuunnan mukaisesti kääntämällä se käyrällä olevan sijaintinsa tangentin suuntaisesti. (Flick 2014.)

$$p = p + t : D \quad (8)$$

p on objektin sijainti käyrällä prosenttiyksikkönä (0…1) alusta loppuun
t on edelliseen ruudunpäivitykseen kulunut aika sekunteina
D on liikkeen kesto käyrän alusta loppuun sekunteina

Hiukkaset keräystehosteelle toteutettiin hyödyntämällä viittä hiukkasjärjestelmää. Kolme järjestelmistä on kiinni käyrää pitkin liikuteltavassa objektissa. Kaksi hiukkasjärjestelmistä on kiinni liikkeen loputtua aktivoitavassa tähdessä.

Käyrällä liikkuvista hiukkasjärjestelmistä ensimmäinen erittää vain yhden hiukkasen, joka ei itse liiku, mutta liikkuu käyrällä liikkuvan emo-objektinsa mukana. Tämä hiukkanen jättää liikkuessaan peräänsä sinistä vanaa, joka muuttuu hiljalleen keltaiseksi. Toinen liikkuvista järjestelmistä erittää pieniä tähtiä liikkeen aikana liikettä vastakkaiseen suuntaan. Tähdet pyörivät ja pienenevät lopulta kadoten. Kolmas järjestelmä erittää pieniä ellipsin muotoisia hiukkasia liikkeen vastakkaiseen suuntaan. Nämä pienet ja melko huomaamattomat hiukkaset tuovat tehosteeseen vauhdin tuntua. Kuvassa 23 on pelikuva liikkeessä olevasta valmiista keräilytehosteesta.



Kuva 23. Pelikuva liikkeessä olevasta keräilytehosteesta.

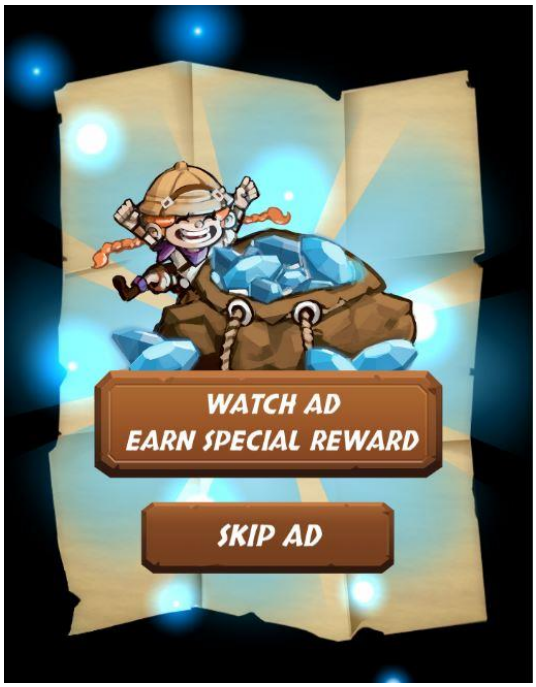
Kun tehoste saavuttaa käyrän loppupisteen, käyrällä liikkuneet hiukkasjärjestelmät lopettavat hiukkasten erittämisen ja tähti asetetaan aktiiviseksi. Samalla hiukkasjärjestelmät, jotka ovat kiinni tähdessä, aktivoituvat. Ensimmäinen järjestelmä erittää suuren määrän erikokoisia tähtiä ympäriinsä. Tähdet pienenevät ja pyörivät, kunnes lopulta katoavat. Toinen järjestelmä on niin sanottu alihiukkasjärjestelmä, joka on kiinni jokaisessa ensimmäisen järjestelmän erittämässä tähtihiukkasessa. Alihiukkasjärjestelmä erittää erikokoisia pieniä valkoisia tähtiä, jotka liikkuvat sinne tänne ja vilkkuvat. Kuvassa 24 on valmiin keräilytehosteen loppuhuipennus.

Kuva 24. Valmiin keräilytehosteen loppuhuipennus pelissä.

Palkintoruudun tehoste

Palkintoruudussa pelaajalle tarjotaan katsottavaksi mainosta palkintoa vastaan. Ruudussa on kuva pelihahmosta timanttisäkin kanssa, mutta kuva ilman liikettä ei yksin riittänyt korostamaan palkinnon ja mainoksen katsomisen tärkeyttä halutulla tavalla. Palkinnosta ja mainoksen katsomisesta haluttiin tehdä houkuttelevaa pelaajalle erikoistehosteen avulla. Tehoste luotiin hiukkasjärjestelmän ja yksinkertaisen animaation avulla.

Kuvassa 25 on valmis palkintoruudun tehoste. Loistavat siniset ympyrät on toteutettu käyttämällä kahta hiukkasjärjestelmää. Hiukkasjärjestelmät erittävät hiukkasia sijainnistaan satunnaiseen suuntaan xy-tasossa. Eritetyt hiukkaset liikkuvat kohti näytön reunoja pienentyen koko ajan ja lopulta kadoten. Hiukkasten hehku on saatu aikaan käyttämällä additiivista varjostinta, joka summaa hiukkasen ja taustan värit ja saa hiukkasen peittämän kohdan näytöllä näyttämään hehkuvalta. Ainut palkintoruudun hiukkasjärjestelmien välillä oleva ero on hiukkasten tekstuuri.



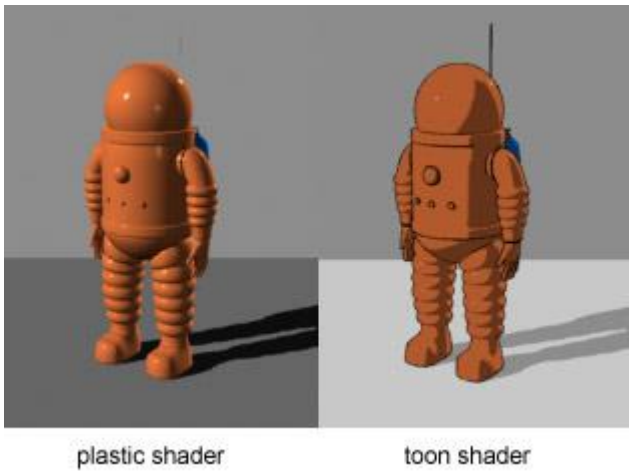
Kuva 25. Pelin palkintoruutu.

Ruudun sininen loiste, joka on timanttisäkin ja paperin välissä, on toteutettiin käyttämällä Unity-pelimoottorin animointityökalua. Animointityökalulla voidaan nauhoittaa mihin tahansa peliobjektiin editorissa tehdyt muutokset. Työkalu toimii siten, että valitaan nauhoitettava objekti, käynnistetään nauhoitus ja tehdään halutut muutokset. Loistetta animoitaessa ainut muutettava muuttuja oli peliobjektin rotaatio, jota muutettiin 360 astetta z-akselin mukaisesti. Kun tallennus on valmis, animaatio voidaan toistaa. Oletuksella animaatiotyökalu muokkaa peliobjektin arvoja lineaarisesti nauhoituksen alkutilasta lopputilaan. Loisteen tapauksessa animaatio muuttaa objektin rotaatiota 360 astetta z-mukaisesti toistuvasti. Loiste toisin sanoen pyörii ympyrää koko ajan. Animaatiotyökalulla on helppoa ja nopeaa luoda yksinkertaisia tehosteita, kuten loisteen pyöriminen, jotka voisi muutoin joutua ohjelmoimaan.

4.3 Varjostimet

Varjostimet ovat tietokoneen grafiikkaprosessorilla suoritettavia ohjelmia, joissa suoritetaan valaistukseen ja renderointitehosteisiin liittyvä koodi. Varjostimia pyritään käyttämään pelituotannossa kaikkeen, mihin niitä voidaan käyttää. Näin voidaan säästää keskusprosessorin aikaa. Kuvassa 26 on esimerkki siitä, miten erilaisiin lopputuloksiin samasta lähtöpisteestä voidaan päästä eri varjostimia hyödyntämällä. Kuvassa

vasemmalla on perinteisempi muovia jäljittelevä varjostin ja oikealla sarjakuvapiirroksen tyyliä tavoitteleva varjostin.



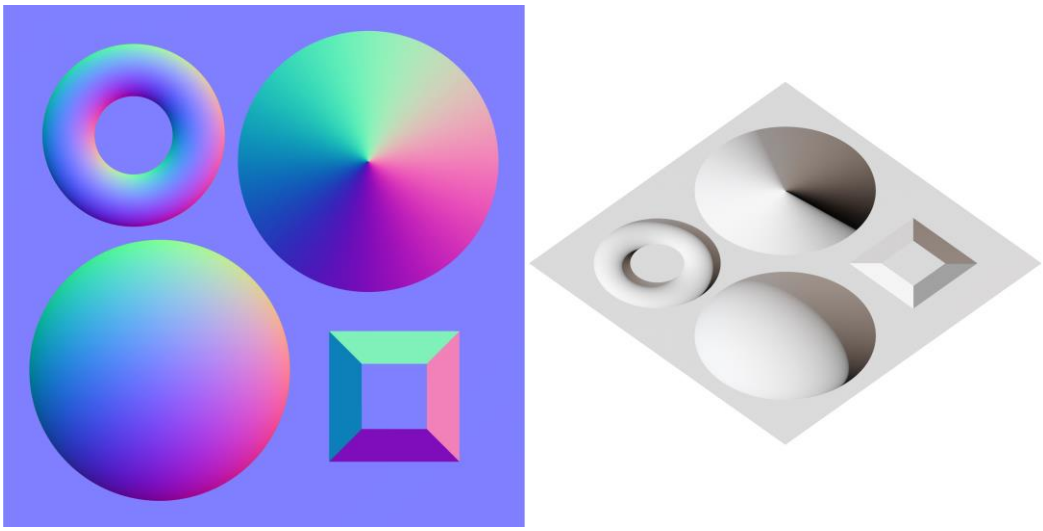
Kuva 26. Erilaisten varjostimien vertailua (Toon shader 2005).

Unity-pelimoottori tukee HLSL-, GLSL- ja Cg-varjostimia. GLSL-varjostinta ei kuitenkaan suositella käytettäväksi sen alustarajoitteiden takia (Unity Manual – GLSL Shader programs 2018). Pelimoottorissa on mahdollista käyttää myös Unityn omia Surface-varjostimia. Surface-varjostimet ovat oikeammin koodigeneraattoreita HLSL-varjostimille.

Kaikki edellä mainitut varjostimet tukevat kärkipiste- ja pikselivarjostimia. Kärkipistevarjostin on ohjelma, joka suoritetaan jokaiselle 3D-mallin kärkipisteelle. Usein kärkipistevarjostimissa vain siirretään mallin piste leiketilaan, jota grafiikkaprosessori käyttää rasteroidakseen kuvan näytölle. Kärkipistevarjostimessa on kyllä mahdollista tehdä kärkipisteelle lähes mitä vain, mutta se ei yleensä ole suositeltua. Varjostimien koodi suoritetaan vasta pelikoodin jälkeen, joten lopputulokset eivät välttämättä olisi pelitilanteen kannalta toivottuja. Pikselivarjostin suoritetaan jokaiselle mallin pikselille, ei näytön pikselille. Pikselivarjostimia suoritetaan usein miljoonille pikseleille, minkä takia ne tulee optimoida hyvin, jottei ruudunpäivitysnopeus laskisi. (Unity Manual – Vertex and fragment shader examples 2018.)

Insinöörityön tehosteita varten toteutettiin muutama yksinkertainen varjostin, jotka toteutettiin Surface-varjostimina. Valaistusta varten toteutettiin varjostin, joka laskee varjot 2D-objekteille ja käyttää normaalikarttoja valaistuksen tehostamiseen. Normaalikarttojen avulla huijataan valon taittumista objektin pinnasta. Normaalikartat ovat tekstuureita, jotka sisältävät informaation pinnan normaalin suunnasta RGB-värinä. RGB-värin

punainen (R) vastaa koordinaatiston X-komponenttia, ja vastaavasti vihreä (G) ja sininen (B) vastaavat Y- ja Z-komponentteja. Kuvassa 27 on esimerkki normaalikartasta ja sen avulla saadusta lopputuloksesta neliön muotoiselle tasolle. (Lammers 2013: 71–75.)



Kuva 27. Normaalikartta ja sen avulla aikaansaatu vaikutelma valon taittumisesta tasolta (Normal map example 2013).

Toinen varjostin toteutettiin kuviollisia laattoja varten, jotta laatan kuvio saatiin hehkumaan. Hehku saatiin aikaan antamalla varjostimelle vaalea tekstuuri, joka on kuvion muotoinen ja muuten läpinäkyvä. Tekstuurin määrittämä valkea alue kerrotaan varjostimessa halutulla värillä, ja hehku on valmis. Hehkun voimakkuutta voidaan vaihdella pelin aikana, jotta hehkuun saadaan eloa. Hehku poistetaan pelaajahahmon kävellessä kuviolaatan yli, mikä indikoi kuviolaatan aktivoitumista. Kuvassa 28 on hehkun aikaansaantiin tarvittavat tekstuurit ja niiden avulla saatu lopputulos.

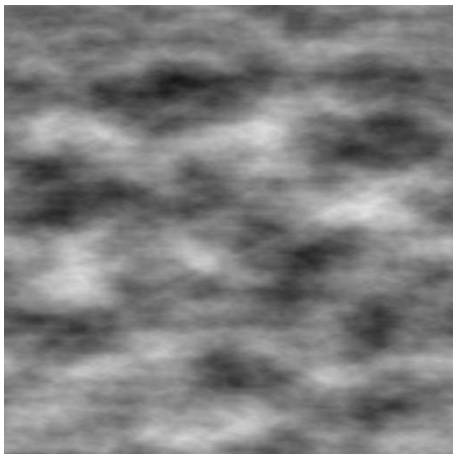
Kuva 28. Hehkuesimerkki. Vasemmalta oikealle hehkutekstuuri, laatan tekstuuri ja lopputulos.

Muut toteutetut varjostimet ovat hyvin samankaltaisia kuin aiemmin kuvaillut, joten niitä ei ole aiheellista käydä läpi. Poikkeuksena on hiekkamyrskytehostetta varten toteutettu varjostin, joka käydään läpi luvussa 4.4.

4.4 Hiekkamyrsky

Hiekkamyrskytehosteen on tarkoitus korostaa pelaajalle pelin kartalta valittavien kohteiden lukittua tilaa. Hiekkamyrskyn lisäksi lukittua tilaa korostetaan kohteiden grafiikan värittömyydellä ja normaalien käyttöliittymäpainikkeiden puuttumisella.

Hiekkamyrskytehoste toteutettiin UV-vierityksen ja hiukkastehosteen avulla. UV-vierityksellä tarkoitetaan tekstuurin koordinaattien muuttamista. Tällä tavalla saadaan luotua tunne liikkeestä ilman, että mallin tai kuvan geometriaa tarvitsee muuttaa. UV-vieritystä tehtäessä on yleensä olennaista, että vieritettävä tekstuuri on saumattomasti toistuva, jotta sitä liikutettaessa kuvan reunat eivät ole havaittavissa. Kuvassa 29 on saumattomasti toistuva tekstuuri, jota käytettiin hiekkamyrskytehosteen luonnissa.



Kuva 29. Saumattomasti toistuva tekstuuri.

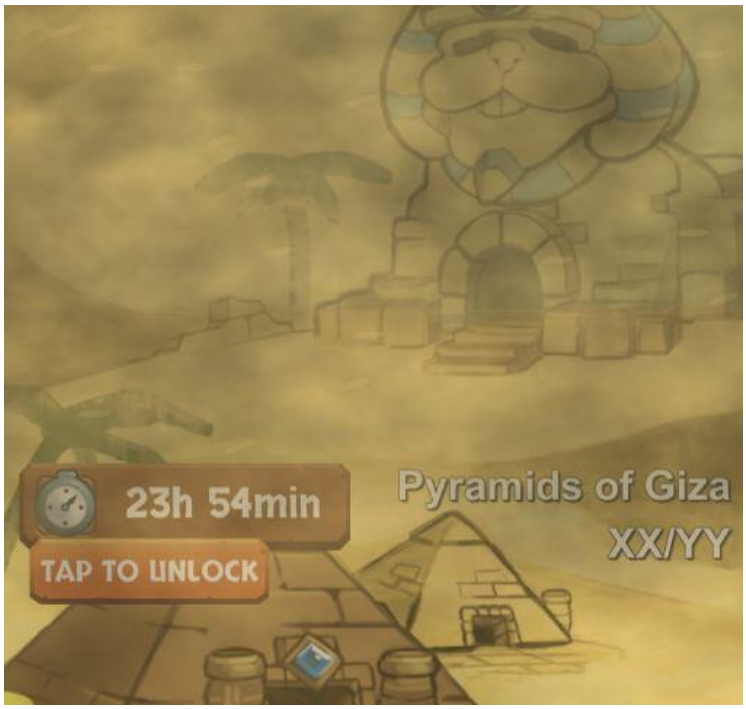
UV-vieritys toteutettiin varjostin koodissa kaavan 9 avulla. Toteutetulle varjostimelle voidaan antaa syötteenä väri, neljä tekstuuria ja kolmelle ensimmäiselle tekstuurille vieritysnopeus x- ja y-akselien suuntaisesti. Varjostin piirtää kolme ensimmäistä tekstuuria päällekkäin halutunvärisinä ja muuttaa niiden UV-koordinaatteja annetuilla nopeuksilla. Neljäs tekstuuri on alfatekstuuri, jonka alfakanavan avulla varjostin määrittää kappaleen läpinäkyvyyden. Kun useaa tekstuuria liikutellaan eri nopeuksilla, tehosteesta saadaan näyttävämpi kuin yhdellä tekstuurilla. Vieritysnopeus täytyy skaalata edelliseen

ruudunpäivitykseen kuluneella ajalla, sillä jos näin ei tehdä, vieritysnopeus vaihtelee käytetyn laitteen tehokkuuden mukaan. (Lammers 2013: 55–58.)

$$\overrightarrow{UV} = \overrightarrow{P_{UV}} + v \cdot t \quad (9)$$

$\overrightarrow{P_{UV}}$ on kärkipisteen UV-koordinaattivektori
v on nopeus, jolla koordinaatteja halutaan muokata
t on edelliseen ruudunpäivitykseen kulunut aika

Hiukkastehosteen avulla hiekkamyrskyyn toteutettiin melko huomaamattomia hiekanjyviä, jotka liikkuvat nopeasti ruudun poikki hiekkamyrskyn suuntaisesti oikealta vasemmalle. Hiukkasten koko kerrotaan niiden nopeudella, jotta ne näyttäisivät liikkuvan kameran ohi suurella nopeudella. Hiukkastehoste toteutettiin tehostamaan tunnelmaa hiekkamyrskystä. Ilman hiukkastehostetta tehosteen voi kuvitella savuna. Kuvassa 30 on valmis hiekkamyrskytehoste pelin valikossa.



Kuva 30. Valmis hiekkamyrskytehoste pelin kohdevalikossa.

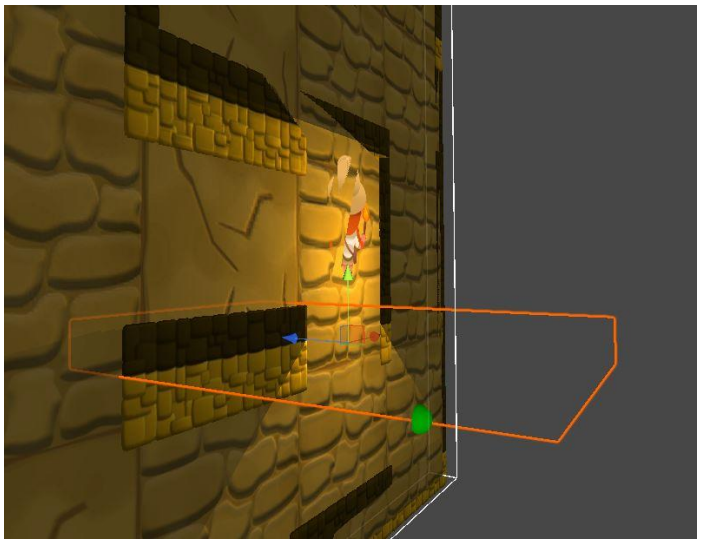
4.5 Valaistus

Trail of Relics -pelin valaistuksella haluttiin luoda tunnelma maanalaisista paikoista, joissa kukaan ei ole käynyt vuosiin. Ainoa valonlähde tällaisissa paikoissa olisi pelihahmon pitelemä soihtu. Pelissä on soihdun lisäksi yleinen ympäristövalaistus, joka valaisee kaikkia peliobjekteja samalla tavalla koko ajan. Ilman ympäristövalaistusta kaikki soihdun valaiseman alueen ulkopuolella olisi mustaa.

Trail of Relics on niin sanottu 2.5D-peli, eli peli antaa vaikutelman kolmiulotteisuudesta, vaikka kaikki grafiikka on oikeasti kaksiulotteista, ja pelin liike tapahtuu kahdessa ulottuvuudessa (Create 2D and 3D games in Unity). Kaksiulotteisuuden vuoksi haluttua soihtuvalaistusta ei voitu lisätä peliin niin vain, sillä varjojen luomiseen tarvitaan kolmiulotteista geometriaa. Kuvassa 31 on vasemmalla yksi pelin tasoista, jossa kaikki objektit ovat kaksiulotteisia, ja oikealla sama taso, johon on lisätty kolmiulotteisia objekteja luomaan varjoja.

Kuva 31. Valaistus ilman kolmiulotteista geometriaa (vasemmalla) ja sen kanssa (oikealla).

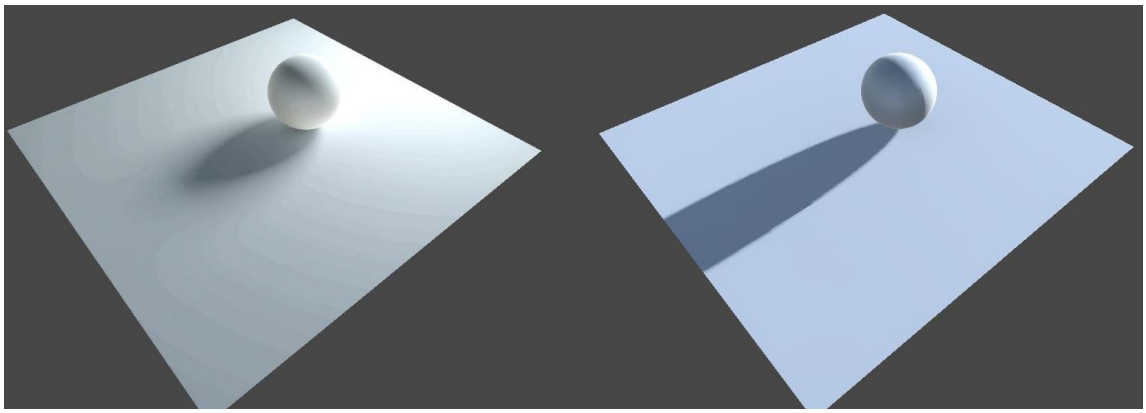
Kolmiulotteinen geometria, joka luo kuvassa 31 (oikealla) varjot, toteutettiin lisäämällä pelin seinille näkymätön suorakulmaisen särmiön muotoinen 3D-malli, joka näkyy kuvassa 32. Tällä tavalla saatiin aikaan varjot, jotka luovat haluttua tunnelmaa ja tuntuvat nopealla tarkastelulla realistisilta, vaikkeivat sitä todellisuudessa olekaan.



Kuva 32. Jokaiselle pelin seinälle on luotu vastaava näkymätön geometria.

Liekkiä muistuttava valosta saatiin laskemalla valon intensiteetti ja kantama jokaisella ruudunpäivityksellä uudestaan sinifunktion avulla sekä kertomalla tulos satunnaismuuttujalla. Satunnaismuuttujaa käyttämällä valo saatiin käyttäytymään arvaamattomasti, kuin oikea liekki.

Soihtu on dynaaminen valo, mikä tarkoittaa sitä, että sen sijainti ja muut asetukset voivat muuttua pelin aikana. Dynaamisuuden vuoksi soihdun luoma valaistus pitää laskea jokaisella ruudunpäivityksellä uudestaan. Dynaaminen valaistus vaatii erittäin paljon laskentatehoa, minkä takia sitä tulee mobiilipeleissä käyttää säästeliäästi, jos ollenkaan (Unity Manual – Graphics Methods 2018). Aluksi soihdun valo vaati liikaa laskentatehoa, ja pelin ruudunpäivitysnopeus laski niin paljon, että selvää nykimistä oli havaittavissa. Vaadittua laskentatehoa saatiin laskettua määrittämällä varjojen piirtoetäisyys ja resoluutio pienemmiksi ja käyttämällä vain teräviä varjoja. Kuva 33 havainnollistaa pehmeiden ja terävien varjojen eroja hyvin kärjistetysti.



Kuva 33. Ääriesimerkit pehmeistä (vasemmalla) ja terävistä (oikealla) varjoista. Esimerkit on toteutettu erilaisilla valonlähteillä.

Muutakin valaistusta suunniteltiin käytettäväksi pelissä tehostamaan pelin tapahtumia, kuten pelin tason oven avautumista, mutta suunnitelmista kuitenkin luovuttiin. Dynaamiset valot olisivat vaatineet liikaa laskentatehoa mobiililaitteilta, eikä ennalta laskettua valaistusta voitu käyttää. Kaksiulotteisille objekteille, jotka käyttävät Unity-pelimoottorin Sprite Renderer -komponenttia, ei voida oletuksena laskea valaistusta ennalta (Unity Manual: Lightmapping: Getting Started 2018). Tästä syystä kaksiulotteisten objektien valaistus kannattaa yleisesti piirtää suoraan grafiikkaan.

5 Tulosten arviointi

Insinöörityön tuloksena valmistui vähintään demo- eli testiversio kaikista peliin vaadituista erikoistehosteista. Täysin valmiista erikoistehosteista ei voida puhua, sillä työn erikoistehosteiden ulkoasu ei ole loppuun asti hiottu eikä tehosteita ole optimoitu niin hyvin kuin mahdollista. Pelin kehityksen kannalta optimointia tärkeämpää oli saada kaikki tehosteet toteutettua, jotta pelin testiversio olisi mahdollisimman lähellä lopputuotetta. Pelitestien palaute olisi täten monipuolisinta ja hyödyllisintä jatkokehityksen kannalta. Valmistuneiden tehosteiden toimivuus saatiin todettua pelitesteissä, joissa pelin interaktiot olivat pelaajille helppotajuisempia kuin aikaisemmin.

Luvussa 4 esiteltyjen tehosteiden lisäksi työssä toteutettiin muitakin tehosteita. Pääasiassa nämä muut tehosteet ovat hiukkastehosteita tai erittäin yksinkertaisia varjostimia. Näitä tehosteita ei käydä työssä läpi yksityiskohtaisesti siitä syystä, että ne ovat tekniikaltaan todella samankaltaisia kuin läpikäydyt tehosteet. Esimerkiksi kuvan 34 hiukkastehoste, joka aktivoituu, kun pelaajahahmo kävelee kuviollisen laatan yli, jakaa paljon samoja elementtejä keräilytehosteen kanssa.

Kuva 34. Laatan aktivointitehoste.

Vertailtaessa valmistuneita tehosteita luvun 3 referenssitehosteisiin voidaan todeta referenssien ulkoasujen vaikuttaneen valmistuneiden tehosteiden ulkoasuihin. Referenssitehosteiden ja valmistuneiden erikoistehosteiden teknisiä toteutuksia ei ikävä kyllä voida vertailla, sillä referenssitehosteiden tekniikka ei ole tiedossa. On kuitenkin hyvin todennäköistä, että toteutetulla reittitehosteella voitaisiin luoda kuvan 7 kaltainen reittitehoste,

ja hiekkamyrskytehosteen varjostimella olisi varmasti mahdollista luoda kuvan 13 tyyppinen hiekkamyrskytehoste. Niiden toteuttaminen luoduilla tekniikoilla vaatisi suurin piirtein vain tekstuurien vaihtamista.

Erikoistehosteiden lisäksi työn tuloksena valmistui uudelleenkäytettävää materiaalia, kuten varjostimia, tekstuureja ja Bézier-käyrätoteutus. Etenkin käyrätoteutus on helposti uudelleenkäytettävissä muissa projekteissa. Käyrätoteutuksen avulla voidaan muun muassa luoda objekteja piirretylle käyrälle, luoda polygonimalleja tai liikuttaa esimerkiksi kameraa käyrää pitkin elokuvamaista kuvaa tavoiteltaessa. Toteutuksen heikkous lienee se, että jos sitä halutaan käyttää johonkin monimutkaisempaan tarkoitukseen, silloin joudutaan varmasti toteuttamaan skriptejä, jotka laajentavat toteutusta. Esimerkiksi reittitehosteen luomiseksi täytyi ohjelmoida laajennus, joka loi käyrän käyttäjän syötteiden perusteella.

Valaistus koettiin ainakin taiteellisesti onnistuneeksi, sillä se sai ajoittain positiivisia kommentteja pelaajilta. Pelaajat eivät tuntuneet huomaavan valaistuksen epärealistisia ominaisuuksia, ellei niistä erikseen huomautettu tai kysytty. Teknisen toteutuksen ei kuitenkaan uskota olevan paras mahdollinen sen ajoittaisten realismi- ja suorituskykyongelmien takia, mutta nopeaan prototyypin luomiseen tekniikka sopii erittäin hyvin.

Pelin kehitystä saatetaan jatkaa, jos peli menestyy riittävän hyvin maailmanlaajuisen julkaisun jälkeen. Tällöin myös työssä esiteltyjä erikoistehosteita jatkokehitettäisiin ja uusia luotaisiin. Jatkokehitys tarkoittaisi suurelle osalle tehosteista tarkkaa optimointia ja näyttävyyden hiomista huippuunsa. Valaistusta ainakin optimoitaisiin ja uusia valoja yritettäisiin lisätä peliin. Valaistustekniikkakin saatettaisiin vaihtaa, jos jollain muulla tekniikalla saataisiin realistisempi lopputulos ilman suuria suorituskykyhaittoja. Vastaavasti jos peli ei menesty, erikoistehosteetkin saavat jäädä nykyiselleen.

6 Yhteenveto

Insinöörityön tavoitteena oli luoda erikoistehosteet Trail of Relics -mobiilipeliin. Erikoistehosteiden tavoitteena oli tehdä pelistä näyttävämpi ja helppotajuisempi pelaajalle. Lisäksi tavoitteena oli perehtyä erilaisiin erikoistehosteiden toteutustekniikoihin ja -menetelmiin.

Työn aikana perehdyttiin usean eri erikoistehosteartistin kirjoittamiin blogeihin ja artikkeleihin. Kirjoituksista omaksuttiin muun muassa alan termistöä, tekniikoita ja käytäntöjä. Artikkeleiden avulla työssä pystyttiin määrittelemään videopelien erikoistehosteiden yleinen tuotantoprosessi sekä erikoistehosteen rakenne ja elinkaari.

Kaikki erikoistehosteet toteutettiin käyttämällä Unity-pelimoottoria ja C#-ohjelmointikieltä. Tekstuurit piirrettiin eri kuvienkäsittelyohjelmilla. Toteutettaessa tehosteita Unity-pelimoottorin toiminnasta saatiin selville uutta liittyen etenkin valaistukseen ja renderointiin. C#-ohjelmointikielestä ja sen muistinhallinnasta opittiin paljon optimoitaessa skriptejä käyttämään vähemmän laskentatehoa ja välimuistia. Varjostimien toteutuksen aikana ymmärrettiin, että lähes kaikki, mitä varjostimien avulla voi tehdä, kannattaa tehdä nimenomaan varjostimilla. Varjostimien tehokas käyttö voi säästää keskusprosessorin aikaa kriittisesti.

Työn tuloksena kaikista vaadituista tehosteista valmistui demo- eli testiversiot, joita tullaan todennäköisesti käyttämään pelin julkaisuversiossa. Julkaisuversion menestyksen perusteella päätetään, aiotaanko peliä jatkokehittää. Tällöin myös tehosteita jatkokehitettäisiin, mikä olisi pääosin ulkoasun hiomista ja koodin optimointia.

Valmiiden tehosteiden lisäksi työn tuloksena syntyi myös uudelleenkäytettävää tekniikkaa. Esimerkiksi kaikkia varjostimia ja Bézier-käyrätoteutusta (ks. luku 4.1) voidaan hyödyntää tulevissa projekteissa. Etenkin käyrätoteutus on hyödynnettävissä muussakin kuin erikoistehostekäytössä. Toteutuksen avulla voidaan esimerkiksi generoida objekteja käyrälle sekä luoda kompleksisia polygonimalleja tai liikuttaa objekteja käyrää pitkin.

Lähteet

Treasure map. 2010. Verkkoaineisto. Wikimedia Commons. <https://commons.wikimedia.org/wiki/File:Treasure_map.svg> 9.4.2010. Luettu 29.6.2018.

Abdillah, Baiquni. 2016. Studying Lighting in Games. Verkkoaineisto. 80 level. <https://80.lv/articles/studying-lighting-in-games/> 18.2.2016. Luettu 5.7.2018.

Bézier-käyrä. 2006. Verkkoaineisto. Wikipedia. <https://id.wikipedia.org/wiki/Berkas:Bezier_curve.svg> 13.5.2006. Luettu 6.7.2018.

Blades of Brim. 2016. Verkkoaineisto. Youtube. <https://www.youtube.com/watch?v=1viimVlz7z0&> 13.11.2016. Luettu 27.8.2018.

Create 2D and 3D games in Unity. Verkkoaineisto. Unity Technologies. <https://unity3d.com/difference-between-2d-and-3d-games> Luettu 25.7.2018.

Cut the Rope: Magic Release Trailer. 2015. Verkkoaineisto. ZeptoLab. <https://www.youtube.com/watch?v=Yoxax1OUOss> 16.12.2015. Luettu 2.7.2018.

de Laat, Sjors. 2018. Learning FX Workflow. Verkkoaineisto. 80 lvl. <https://80.lv/articles/learning-fx-workflow/> 22.8.2018. Luettu 24.8.2018.

Dota 2. 2013. Verkkoaineisto. Valve Corporation. <https://www.youtube.com/watch?v=4rDFMvlnNck> Luettu 29.6.2018.

Europa Universalis IV. 2013. Verkkoaineisto. Paradox Interactive. <www.europauniversalis4.com> 13.8.2013. Luettu 29.6.2018.

Flick, Jasper. 2014. Curves and Splines, making your own path. Verkkoaineisto. Catlike Coding. <https://catlikecoding.com/unity/tutorials/curves-and-splines/> 31.7.2014. Luettu 10.7.2018.

Fortnite. 2017. Verkkoaineisto. Epic Games. <https://www.youtube.com/watch?v=IyWGkmXB_xc> 26.9.2017. Luettu 29.6.2018.

Frisvad, Jeppe Revall; Christensen, Niels Jørgen & Falster Peter. Lighting Effects for Mobile Games. Verkkoaineisto. <https://pdfs.semanticscholar.org/28c8/fd4fe10184e753366a55dc328b72ee5d3a44.pdf> Luettu 5.7.2018.

Gallahan, Nathan. 2006. Retired master sergeant weather new career. Verkkoaineisto. United States Air Force. <http://www.publicaffairs.af.mil/News/Features/Display/Article/143682/retired-master-sergeant-weathers-new-career/> 13.1.2006. Luettu 27.6.2018.

Guerrette, Keith. 2016. The Desired Effect: How Visual Effects Are Essential to Video Game Storytelling. Verkkoaineisto. <https://www.youtube.com/watch?v=rBvXfJqjdc4> 17.2.2016. Luettu 27.6.2018.

Hearhstone. 2014. Verkkoaineisto. Blizzard Entertainment. <https://www.youtube.com/watch?v=YVPwKljGelg> Luettu 3.7.2018.

Heazlewood, James. 2013. Unity Sparks System. Verkkoaineisto. <https://seagullcity.wordpress.com/2013/10/13/unity-sparks-system/> 13.10.2013. Luettu 30.6.2018.

Sandstorm. Verkkoaineisto. <https://www.maxpixel.net/Egypt-Pyramids-Sandstorm-103304> Luettu 3.7.2018.

Hubbell, Nathaniel. 2014. How to Become a Video Game Special Effects Artist. Verkkoaineisto. Game Industry Career Guide. <https://www.gameindustrycareerguide.com/how-to-become-a-video-game-special-effects-artist/> 1.3.2014. Luettu 27.6.2018.

Jevremovic, Stefan. 2018. VFX Staples: Shape, Color, and Motion. Verkkoaineisto. 80 level. <https://80.lv/articles/vfx-staples-shape-color-and-motion/> 17.1.2018. Luettu 27.6.2018.

Kamermans, Mike. 2011. A Primer on Bézier Curves. Verkkoaineisto. GitHub. <https://pomax.github.io/bezierinfo/#introduction> 2011. Luettu 10.7.2018.

Lague, Sebastian. 2018. Curve Editor. Verkkoaineisto. YouTube. <https://www.youtube.com/playlist?list=PLFt_AvWsXl0d8aDaovNztYf6iTChHzrHP> 26.2.2018. Luettu 16.7.2018.

Lammers, Kenny. 2013. Unity Shaders and Effects Cookbook. Packt Publishing.

Normal map example. 2013. Normal map example with scene and result. Verkkoaineisto. <https://commons.wikimedia.org/wiki/File:Normal_map_example_with_scene_and_result.png> 8.4.2013. Luettu 6.8.2018.

Okun, Jeffrey A. & Zwerman Susan. 2010. The VES Handbook of Visual Effects: Industry Standard VFX Practices and Procedures. USA: Focal Press.

Ordoñez, Francisco Garcia-Obledo. 2017. VFX for Games Explained. Verkkoaineisto. 80level. <https://80.lv/articles/vfx-for-games-explained/> 11.5.2017. Luettu 28.6.2017

Romanowska, Wirginia. 2017. VFX Production for AAA Video Games. Verkkoaineisto. 80 level. <https://80.lv/articles/the-secrets-of-vfx-production-for-doom/> 25.5.2017. Luettu 25.6.2018.

Toon shader. 2005. Toon-shader. Verkkoaineisto. Wikimedia Commons. <https://commons.wikimedia.org/wiki/File:Toon-shader.jpg> 5.6.2005. Luettu 6.8.2018.

Passage souterrain à Oudna. 2014. Jeux d'Ambre a Oudna. Verkkoaineisto. Wikimedia Commons. <https://commons.wikimedia.org/wiki/File:Jeux_d%27Ambre_a_Oudna.JPG> 29.3.2014. Luettu 3.7.2018.

Unity Manual – GLSL Shader programs. 2018. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/SL-GLSLShaderPrograms.html> Luettu 6.8.2018.

Unity Manual – Graphics Methods. 2018. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/MobileOptimizationGraphicsMethods.html> Luettu 25.7.2018.

Unity Manual – Lightmapping: Getting Started. 2018. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/Lightmapping.html> 28.3.2018. Luettu 24.7.2018.

Unity Manual – Line Renderer. 2017. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/class-LineRenderer.html> 31.5.2017. Luettu 6.7.2018.

Unity Manual – Vertex and fragment shader examples. 2018. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/SL-VertexFragmentShaderExamples.html> Luettu 6.8.2018.

Unity Manual – What is a Particle System. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/PartSysWhatIs.html> Luettu 30.6.2018.

Xuexiang, Li & Junxiao, Xue. 2014. Complex Quadratic Bézier Curve on Unit Circle. Verkkoaineisto. <https://www.atlantis-press.com/php/download_paper.php?id=12403> 2014. Luettu 12.7.2018.

Pisteiden lasku tietyn välimatkan välein kuutiolliselle Bézier-yhdistelmäkäyrälle

```
List<Vector3> controlPoints = new List<Vector3>();
List<float> curveLengths = new List<float>();
public List<Vector3> CalculateEvenlySpacedPoints(float spacing, float resolu-
tion = 1, int startCurve = 0, int endCurve = -1)
{
    if (endCurve == -1)
        endCurve = CurveCount;
    if (endCurve < startCurve)
        throw new Exception("endCurve value can't be smaller than
startCurve");
    if (startCurve > CurveCount)
        throw new Exception("startCurve is bigger than curvecount");

    int numControlPoints = (endCurve - startCurve) * 3 + 1;
    if (controlPoints.Capacity < numControlPoints)
        controlPoints = new List<Vector3>(numControlPoints);
    for (int i = 0; i < numControlPoints; ++i)
    {
        controlPoints.Add(GetControlPoint(i + startCurve * 3));
    }

    float estimatedSplineLength = 0;
    int numCurves = endCurve - startCurve;
    if (curveLengths.Capacity < numCurves)
        curveLengths = new List<float>(numCurves);

    // Calculate estimatedSplinelength
    for (int i = 0; i < controlPoints.Count - 1; i += 3)
    {
        Vector3 p0 = controlPoints[i],
            p1 = controlPoints[i + 1],
            p2 = controlPoints[i + 2],
            p3 = controlPoints[i + 3];

        float chord = Vector3.Distance(p0, p3);
        float controlNetLength = Vector3.Distance(p0, p1) + Vector3.Dis-
tance(p1, p2) + Vector3.Distance(p2, p3);
        float estimatedCurveLength = (chord + controlNetLength) / 2f;

        curveLengths.Add(estimatedCurveLength);
        estimatedSplineLength += estimatedCurveLength;
    }

    #region EvenlySpacedPoints declaration

    if (EvenlySpacedPoints == null || EvenlySpacedPoints.Capacity <
Mathf.RoundToInt(estimatedSplineLength / spacing))
        EvenlySpacedPoints = new List<Vector3>(Mathf.RoundToInt(estimated-
SplineLength / spacing));

    EvenlySpacedPoints.Clear();
    EvenlySpacedPoints.Add(points[startCurve * 3]);
    Vector3 previousPoint = points[startCurve * 3];

    #endregion

    float distanceSinceLastEvenPoint = 0;
    for (int i = 0; i < curveLengths.Count; ++i)
    {
        int j = i * 3;
        int divisions = Mathf.CeilToInt(curveLengths[i] * resolution * 10);
        float t = 0;
```

```
        while (t <= 1)
        {
            t += 1f / divisions;
            Vector3 pointOnCurve = Bezier.GetPoint(controlPoints[j], con-
trolPoints[j+1], controlPoints[j+2], controlPoints[j+3], t);
            distanceSinceLastEvenPoint += Vector3.Distance(previousPoint,
pointOnCurve);

            Vector3 normalizedDir = (previousPoint - pointOnCurve).normalized;
            while (distanceSinceLastEvenPoint >= spacing)
            {
                float overshootDistance = distanceSinceLastEvenPoint - spac-
ing;
                Vector3 newEvenlySpacedPoint = pointOnCurve + normalizedDir *
overshootDistance;
                EvenlySpacedPoints.Add(newEvenlySpacedPoint);
                distanceSinceLastEvenPoint = overshootDistance;
                previousPoint = newEvenlySpacedPoint;
            }

            previousPoint = pointOnCurve;
        }
    }

    curveLengths.Clear();
    controlPoints.Clear();

    return EvenlySpacedPoints;
}
```